UNIVERSIDAD POLITÉCNICA
DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INFORMÁTICOS

MÁSTER UNIVERSITARIO EN INGENIERÍA DEL SOFTWARE –
EUROPEAN MASTER ON SOFTWARE ENGINEERING

Categorization of Software Project Management Anti-Patterns

Master Thesis

Pedro Dias e Silva

Madrid, June 2014

This thesis is submitted to the ETSI Informáticos at Universidad Politécnica de Madrid in partial fulfillment of the requirements for the degree of Master of Science on Software Engineering.

Master Thesis
Master Universitario en Ingeniería del Software – European Master on Software Engineering
Thesis Title: Categorization of Software Project Management Anti-Patterns
Thesis no: EMSE-2014-02
June, 2014

Author: Pedro Dias e Silva
Bachelor of Science in Computer Science
University of Washington

Supervisor:

Ana M. Moreno
Ph.D. in Computer Science
Universidad Politécnica de Madrid

School of Computing
Universidad Politécnica de Madrid

Co-supervisor:

Lawrence J. Peters
Ph.D. in Engineering Management
California Coast University

Software Project Manager and Consultant
Software Consultants International Limited

ETSI Informáticos
Universidad Politécnica de Madrid
Campus de Montegancedo, s/n
28660 Boadilla del Monte (Madrid)
Spain

Table of Contents

Index of Tables

Index of Figures

1 Introduction

In the last decades, software systems have become an intrinsic element in our daily lives. Software exists in our computers, in our cars, and even in our refrigerators. Today's world has become heavily dependent on software and yet, we still struggle to deliver quality software products, on-time and within budget.

McConnell (2004) reports that cancelled software projects impose an approximately $40 billion drain on the United States economy. He also provides a number of examples where the software delivered was either over-budget or of lower quality than required, leading to enormous losses. Just to list a few examples:

- U.S. Internal Revenue Service (IRS) software modernization project: $50 billion per year in lost revenue.
- U.S. Federal Aviation Administration (FAA) automation system: planned budget overrun by $3 billion.
- Denver International Airport baggage handling system: delays costing as high as $1.1 million per day.

When searching for the causes of such alarming scenario, we find concurrent voices pointing to the role of the project manager. Brown et al. (2000) state that, "The primary cause of software development failure is the lack of appropriate project management." Weinberg (1994) and McConnell (1998) have also expressed similar points of view.

But what is project management and what makes it so challenging?

The first part of the question might be best answered in the words of Peters (2008): "Software project management is not software development and all that development involves. It is all about coordinating and directing software development activities by applying proven engineering management methods and principles that have been adapted to software engineering. It also involves leadership; effective software project managers must both manage and lead."

The second part of the question requires a deeper analysis of why software project managers have been largely ineffective. Answering this question might assist current and future software project managers in avoiding, or at least effectively mitigating, problematic scenarios that, if unresolved, will eventually lead to additional failures. Laplante et al. (2005) argue that, "While it is certainly useful to study the successful ways people solve problems, the old adage that we learn form our mistakes suggests that studying failures might be even more fruitful."

This is where anti-patterns come into play and where they can be a useful tool in identifying and addressing software project management failure. Unfortunately, anti-patterns are still a fairly recent concept, and thus, available information is still scarce and loosely organized.

This thesis will attempt to help remedy this scenario. The objective of this work is to help organize existing, documented software project management anti-patterns by answering our two research questions:

- What are the different anti-patterns in software project management?
- How can these anti-patterns be categorized?

In order to answer these questions properly, in this work we will first cover some basic concepts of anti-patterns, including definition and structure. This information will help us decide what anti-patterns should be considered as we will know what to look for. Next, we will look at the different software project management anti-patterns in the literature and seek to consolidate them into a single list, which should provide an answer to our first research question. We will then attempt to categorize this consolidated list of anti-patterns, answering our second research question, and we will end this work with a discussion of the results obtained in our analysis.

Consequently, this document is structured in the following chapters:

- Chapter 1: *Introduction* contains the introduction with the motivation of our work.
- Chapter 2: *Basic Concepts of Anti-Patterns* explains what software project management anti-patterns are, providing definition and structure.
- Chapter 3: *Anti-Patterns in Literature* examines the software project management anti-patterns found in our research.
- Chapter 4: *Consolidated Anti-Pattern List* presents a consolidated list of software project management anti-patterns.
- Chapter 5: *Anti-Pattern Categorization* categorizes the anti-pattern in the consolidated list, and explains the criteria used in the categorization.
- Chapter 6: *Conclusions* contains the conclusions with an analysis of the results from Chapter 5 and a discussion of how these results could be relevant to software project managers.
- Chapter 7: *References* lists the references used in this work.

2 Basic Concepts of Anti-Patterns

2.1 Introduction

In this chapter we will describe the central topic of this thesis: software project management anti-patterns. We will present the definition of anti-patterns, a brief history of where anti-patterns came from, and the anti-pattern structure we will use in this work. We will end the chapter with the definition of software project management anti-patterns, a brief discussion about them, including how they can be useful to software project managers.

2.1.1 Anti-Pattern Definition

The concept of anti-patterns is leveraged from the principle of using patterns to specify an effective solution and learning from your own and other people's mistakes. "An anti-pattern is a literary form that describes a common response to a problem that generates decidedly negative consequences" [Brown et al., 1998]. Brown et al. use the consequences of this response to differentiate patterns from anti-patterns, and "whether the response leads to positive or negative consequences depends on the context and the execution of the pattern. If the context implies positive outcomes, it's a pattern. If the context implies negative consequences, it's an anti-pattern" [Brown et al., 1998].

Anti-patterns examine the causes, symptoms, and consequences of dysfunctional approaches to problem solving and offer a re-factored solution that provides a successful way of overcoming this dysfunction.

Brown et al. (2000) illustrate the use of anti-patterns with an example,

> We can predict with reasonable accuracy the time we will leave for work in the morning. This is achieved by instituting a process, with known relationships to both time and performance. We rely on technology (alarm, clock, car) and personnel (us, our spouse) to achieve the process. However, we cannot predict the one morning our cars won't start. However, once we know that our cars won't start, we can re-factor a solution knowing the variables at that moment. For example, identifying that the fuel gauge shows empty means that the re-factored solution to the problem is to fill the car with gasoline.

Anti-patterns can have two basic forms: isolated anti-patterns, and interacting anti-patterns of hierarchical or sequential nature. The first can be promptly resolved when identified because it is stand-alone with no hidden causes. The latter is also known as *anti-pattern collision* [Brown et al., 2000] where more than one anti-pattern occur simultaneously; interacting anti-patterns are significantly more difficult to solve as the overlapping of anti-patterns is likely to include several hidden causes.

According to Brown et al. (1998), the purpose of anti-patterns is "not to focus on dysfunctional software practices. Rather, the purpose is the development and implementation of strategies to fix the problems that arise, and to build new awareness that enables you to enhance your success."

2.1.2 Anti-Pattern History

The idea of anti-patterns was initially used for software design and began almost in parallel with that of software design patterns, introduced by Gamma et al. (1994). Credit is often given to Michael Akroyd for the first formal use of the term anti-pattern. In 1996, he presented a paper entitled "AntiPatterns: Vaccinations against Object Misuse" [Akroyd, 1996] at the Object World West conference. However, prior to Akroyd, similar work on identifying and solving dysfunctional behaviors had already been documented by Frederick Brooks [Brooks, 1979], Bruce Webster [Webster, 1995], James Coplien [Coplien, 1995], and Andrew Koenig [Koenig, 1995]. Brown et al. expanded the scope of anti-patterns to include software project management anti-patterns [Brown et al., 1998 and 2000], and Laplante et al. expanded it even further to include environmental, or cultural, anti-patterns [Laplante et al., 2005].

2.1.3 Anti-Pattern Structure

Brown et al. (1998) state that, "A properly documented anti-pattern describes a general form, the primary causes which led to the general form; symptoms describing how to recognize the general form; the consequences of the general form; and a re-factored solution describing how to change the anti-pattern into a healthier situation; without a template, the anti-pattern is just a person's unstructured prose."

It is thus important to use some type of template or structure to document anti-patterns. There are several anti-pattern templates used by different authors and organizations. For the purpose of this work, we have created a simplified adaptation of the template used by Brown et al. [Brown et al., 1998] as we do not require all of the information provided. The following anti-pattern structure will be used in this work:

Anti-Pattern Structure:

- Anti-Pattern Name
- Also Known As (optional)
- Re-Factored Solution Name (optional)
- Re-Factored Solution Type (optional)
- Root Causes (optional)
- Anecdotal Evidence (optional)
- Short Description

Hence, the minimum acceptable information for an anti-pattern to be considered in this work is the anti-pattern name, and a description of the problematic scenario characterized by the anti-pattern. Anti-patterns that do not meet these minimum criteria have been discarded from consideration in this work.

2.1.4 Software Project Management Anti-Patterns

In the context of software project management, Brown et al. (2000) state that, “Anti-patterns are prevalent, recurring software project management roadblocks to successful delivery.” For the purpose of this work, we will use a similar definition in which a software project management anti-pattern is a recurring negative management practice that may or may not have a re-factored solution.

Brown et al. (1998) argue that, “Anti-patterns may be the result of a manager or developer not knowing any better, not having sufficient knowledge or experience in solving a particular type of problem, or having applied a perfectly good pattern in the wrong context.” Laplante et al. (2005) add that in some cases, anti-patterns “might just be tolerating or amplifying the debilitating behavior of another team member, at other times it might be a misguided, incompetent, or even malicious supervisor.”

There can be various causes and explanations for management anti-patterns. Brown has identified eight major root causes for poor management practices, based on the “seven deadly sins” [Brown et al., 1998]:

- **Haste:** Hasty decisions lead to compromises in software quality.
- **Apathy:** Not caring about solving known problems.
- **Narrow-mindedness:** Refusal to practice solutions that are otherwise widely known to be effective.
- **Sloth:** “Healthy sign” of a lazy developer or manager, who makes poor decisions based upon an “easy answer.”
- **Avarice:** Greed can take many forms, but it leads to inappropriate software development decisions.
- **Ignorance:** Intellectual sloth; the result of failing to seek understanding.
- **Pride:** One of the sins of pride is the not-invented-here syndrome, where one does not take advantage of existing solutions that have been proven effective.
- **Responsibility:** The universal cause.

We will refer back to these root causes in the next sections.

3 Anti-Patterns in Literature

3.1 Introduction

In this chapter we will examine all of the software project management anti-patterns found in our research.

Unfortunately, we were not able to find any academic papers with in-depth analysis of anti-patterns. We performed an extensive academic literature search, looking in the main search databases for related keywords, with no success. The only relevant paper found [Stamelos, 2009] merely contained a list of anti-pattern names. In total, we have searched for anti-pattern literature in the following databases.

- Institute of Electrical and Electronics Engineers (IEEE): http://ieeexplore.ieee.org
- Association for Computing Machinery (ACM): http://portal.acm.org
- Web of Knowledge: http://apps.webofknowledge.com
- Directory of Open Access Journals (DOAJ): http://www.doaj.org
- The Institution of Engineering and Technology (IET): http://www.theiet.org/resources/inspec
- Mendeley: http://www.mendeley.com
- Google Scholar: http://scholar.google.com

We used the following sets of keywords to search for anti-pattern literature. Note that for each individual search, we took one keyword from each set, thus conducting our search with numerous combinations of keywords.

Keyword Set A:

- “anti-pattern”
- “antipattern”
- “anti pattern”
- “malpractice”
- “bad practice”

Keyword Set B:

- “software project management”
- “project management”
- “management”

As stated earlier, our search did not return any relevant results. We then proceeded to search for books on software project management anti-patterns, being able to find three sources: two books by Brown et al. (1998, and 2000) and one book by Laplante et al. (2005). We have also found one online source, the Portland Pattern Repository [c2.com, 2014].

Therefore, in this chapter we will examine four lists of anti-patterns, one list for each source found. As a minimum for each anti-pattern, we will present the anti-pattern name and a brief description of the problematic scenario; this is the minimum information used to consider anti-patterns, as discussed previously in section 2.1.3. Additional information may include other names the anti-pattern is known by, the re-factored solution name, the re-factored solution type, the root causes of the anti-pattern, and anecdotal evidence intended to be humorous. The additional information depends on the available information in each source.

Please note that the information presented in this section is taken directly from the four sources stated earlier. We have decided to use the original author's description of each anti-pattern in order to convey this information as accurately as possible.

3.2 Anti-Patterns in Brown et al. (1998)

We will present in this section seven software project management anti-patterns identified by Brown et al. (1998):

1. **Detailitis Plan:** Excessive planning leading to complex schedules with high level of detail, giving the false perception that the project is fully under control.
2. **Fire Drill:** Months of boredom followed by demands for immediate delivery.
3. **Glass Case Plan:** Lack of tracking and updating of initial plans, assuming the plan is enough.
4. **Irrational Management:** Irrational management decisions, habitual indecisiveness and other negative management practices.
5. **Mushroom Management:** Isolating developers from end users, under the mistaken assumption that requirements are stable and well-understood by both end users and the software team at project inception.
6. **Project Mismanagement:** Lack of proper software project monitoring and controlling.
7. **Warm Bodies:** Assuming developers are interchangeable and that the number of people working on a problem is inversely proportional to development time.

3.2.1 Detailitis Plan

Anti-Pattern Name: Detailitis Plan

Also Known As: Death by Planning

Re-Factored Solution Name: Rational Planning

Re-Factored Solution Type: Process

Root Causes: Avarice, Ignorance, Haste

Anecdotal Evidence:

"We can't get started until we have a complete program plan."

"The plan is the only thing that will ensure our success."

Short Description:

Excessive planning for software projects leads to complex schedules that cause downstream problems.

Sometimes the solution to effective delivery is regarded as a high degree of control via a continuous planning exercise that involves most of the senior developers, as well as the managers. This approach often evolves into a hierarchical sequence of plans, which show additional (and unnecessary) levels of detail. The ability to define such a high level of detail gives the perception that the project is fully under control. In reality, software projects are rarely fully under control.

3.2.2 Fire Drill

Anti-Pattern Name: Fire Drill

Re-Factored Solution Name: Sheltering

Anecdotal Evidence:

"Wait until management is desperate, and they will accept anything you give them."

Short Description:

Airline pilots describe flying as "hours of boredom followed by 15 seconds of sheer terror." Many software projects resemble this situation: "Months of boredom followed by demands for immediate delivery." The months of boredom may include protracted requirements analysis, re-planning, waiting for funding, waiting for approval, or any number of techno-political reasons.

3.2.3 Glass Case Plan

Anti-Pattern Name: Glass Case Plan

Also Known As: Death by Planning

Re-Factored Solution Name: Rational Planning

Re-Factored Solution Type: Process

Root Causes: Avarice, Ignorance, Haste

Anecdotal Evidence:

"As long as we follow the plan and don't diverge from it, we will be successful."

"We have a plan; we just need to follow it!"

Short Description:

Often a plan produced at the start of a project is always referenced as if it's an accurate, current view of the project even if it's never updated. This practice gives management a "comfortable view" of delivery before the project starts. However, when the plan is never tracked against, nor updated, it becomes increasingly inaccurate as the project progresses. This false view is often compounded by the absence of concrete information on progress, which often is known only after a critical deliverable slips its schedule.

3.2.4 Irrational Management

Anti-Pattern Name: Irrational Management

Also Known As:

Pathological Supervisor

Short-Term Thinking

Managing by Reaction

Decision Phobia

Managers Playing with Technical Toys

Re-Factored Solution Name: Rational Decision Making

Re-Factored Solution Type: Role, Process

Root Causes: Responsibility

Anecdotal Evidence:

"I wish he'd make up his bloody mind!"

"What do we do now?"

"We better clear this with management before we get started."

"Don't bother asking; they'll just say no."

Short Description:

Habitual indecisiveness and other negative management habits lead to de facto decisions and chronic development crises.

Irrational Management covers a range of commonly occurring software project problems that can be traced back to the personalities of the person(s) running the project. For example, the manager may have obsessive interests in some aspect of the technology or personality limitations that cause them to become ineffective or irrational managers. Irrational management can be viewed as a skewed set of priorities where the manager's personal priorities, no matter how nonsensical, guide the software project in irrational directions.

3.2.5 Mushroom Management

Anti-Pattern Name: Mushroom Management

Also Known As:

Pseudo-Analysis

Blind Development

Anecdotal Evidence:

"Keep your developers in the dark and feed them fertilizer."

"Never let software developers talk to end users."

Short Description:

In some architecture and management circles, there is an explicit policy to isolate system developers from the system's end users. Requirements are passed second-hand through

intermediaries, including architects, managers, or requirements analysts. Mushroom Management assumes that requirements are well understood by both end users and the software team at project inception. It is assumed that requirements are stable.

3.2.6 Project Mismanagement

Anti-Pattern Name: Project Mismanagement

Also Known As: Humpty Dumpty

Re-Factored Solution Name: Risk Management

Re-Factored Solution Type: Process, Role

Root Causes: Responsibility

Anecdotal Evidence:

"Who's running this project?"

"What went wrong? Everything was fine and then suddenly… KABOOM!"

Short Description:

Inattention to the management of software development processes can cause lack of direction and other symptoms. Proper monitoring and control of software projects is necessary to successful development activities. Running a product development is as complex as creating the project plan; and developing software is as complex as building skyscrapers, involving as many steps and processes, including checks and balances. Often, key activities are overlooked or minimized.

This anti-pattern concerns the monitoring and controlling of software project. The timeframe for this occurs after planning activities, and during the actual analysis, design, construction, and testing of the software system. Project mismanagement involves mistakes made in the day-to-day running of a project, assuming planning errors (such as Detailitis Plan and Glass Case Plan) have not been made.

3.2.7 Warm Bodies

Anti-Pattern Name: Warm Bodies

Also Known As:

Deadwood

Body Shop

Seat Warmers

Mythical Man-Month [Brooks, 1979]

Short Description:

Software projects are often staffed with programmers with widely varying skills and productivity levels. Many of these people may be assigned to meet staff size objectives (so-called "warm bodies"). Skilled programmers are essential to the success of a software project. So-called heroic programmers are exceptionally productive, but as few as 1 in 20

have this talent. They produce an order of magnitude of more working software than an average programmer.

Large-scale software projects are prevalent in many industries. These large projects often involve outsourced development and contractual payments based upon labor-hours worked. System requirements always change and increase during development; so there is little risk involved if the project is underbid initially; the contractor can grow the staff to meet inevitable problems and new requirements. The fallacy of adding more staff to an ongoing software project was described by Frederick Brooks in the *Mythical Man-Month* (1979).

3.3 Analysis of Anti-Patterns in Brown et al. (2000)

We will present in this section seven software project management anti-patterns identified by Brown et al. (2000):

1. **Chaos:** Lack of flexible plans and processes.
2. **Micro-Management:** Excessive management involvement in tasks beyond their responsibility.
3. **Myopic Delivery:** Management insisting on original delivery date even when reducing staff or funding.
4. **Process Disintegration:** Failing processes due to an underlying decline in overall cooperation and morale.
5. **Size Isn't Everything:** Arguably the most well-known anti-pattern in software project management. Originally identified by Brooks as the *Mythical Man-Month*, it involves the false assumption that developers are interchangeable and that the number of people working on a problem is inversely proportional to development time.
6. **The Brawl:** Project manager with no leadership or management experience.
7. **The Domino Effect:** Moving critical resources between projects, blurring project boundaries.

3.3.1 Chaos

Anti-Pattern Name: Chaos

Also Known As: The More Things Change, The More Things Change

Re-Factored Solution Name: A Sea of Calm

Re-Factored Solution Type: Role

Root Causes: Haste, Sloth, or Ignorance

Short Description:

Software development plans and processes cannot predict all possibilities that can occur and therefore cannot be followed to the letter to overcome unpredicted problems. A project manager and other managers need to be flexible to ensure that when risks are identified they can be tackled in a flexible, pragmatic, and effective manner. Once chaos starts it is very difficult to restrain. Flexibility of management is the key to resolving this anti-pattern.

3.3.2 Micro-Management

Anti-Pattern Name: Micro-Management

Also Known As:

Unbalanced People

Technology and Process

Attrition 'R' Us

Re-Factored Solution Name: Macro-Management

Re-Factored Solution Type: Role

Root Causes: Pride, Ignorance

Anecdotal Evidence:

"Management don't have any respect for us."

"Not another bloody fire drill! When can we get back to the real work?"

"If we don't tell the developers what to do each day how can we have any control?"

"The developers should do what they're told and if they don't like it, then they should leave!"

Short Description:

Many project managers do not understand how to manage people. They are sometimes excellent technical staff who are promoted or line managers who are given a change of role without necessarily having the required set of skills and experience to manage a software development team.

Project managers that over-manage particular aspects of the project, aspects under the responsibility of subordinates, are likely to do so either because they are weak in that particular area and believe that a micro-focus will mitigate risks or because it is the one skill area that they have.

The range of micro-management can vary significantly; a fire drill is an occasional form of micro-management, while managing daily developer tasks is an extreme form.

Often this leads to the loss of staff, and the managers responsible are surprised because they do not understand the negative effect of their actions on the developers' response.

3.3.3 Myopic Delivery

Anti-Pattern Name: Myopic Delivery

Also Known As: Delivery Zone

Re-Factored Solution Name: Get Out of Town

Re-Factored Solution Type: Process

Root Causes: Ignorance, Pride, Narrow-Mindedness

Anecdotal Evidence:

"What is the difference between management and a terrorist? You can negotiate with the terrorist!"

"I don't care about the schedule anymore, just do it."

Short Description:

Slipped schedules that are faster than can be controlled, generating project compression. Project compression creates a gap in the space-time continuum, which allows the incomprehensible to be accomplished in the minds of management. It causes all logical, pragmatic, rational, reasonable thinking to fly out the window. Management will do anything to meet the delivery date. No one's life is un-expendable. Everyone will suffer. Nothing is more important than the delivery date. This anti-pattern discusses what happens when the schedule slips and management continues to demand the original delivery date.

3.3.4 Process Disintegration

Anti-Pattern Name: Process Disintegration

Also Known As: The Process Club

Re-Factored Solution Name: The Major Technical Collaboration Initiative

Re-Factored Solution Type: Technology

Root Causes: Sloth

Short Description:

The dilemma of how to handle failing processes. Despite its name this anti-pattern is primarily a people-caused anti-pattern that has process consequences. As process advocates in a seeming process-resistant world, we are often faced with the dilemma of just how to handle situations in which a group of professionals is determined to pull defeat from the jaws of victory, and all in the name of principle.

Many times we have stood by helplessly as seasoned professionals managed to disrupt otherwise functional processes, preventing success and souring everyone's attitude. Sometimes these people were just corncobs (i.e., difficult people), and when recognized as such could be appropriately handled. But what is going on when *everyone* seems to have lost it and there is no longer any reasonable hope of continuing without failure? This is where we are likely facing a Process Disintegration anti-pattern.

3.3.5 Size Isn't Everything

Anti-Pattern Name: Size Isn't Everything

Also Known As: How To Have a Baby in One Month with Nine Women

Re-Factored Solution Name: Trusted Cadre, or Rob-the-Rich

Re-Factored Solution Type: Role

Root Causes: Haste, Ignorance, Sloth

Anecdotal Evidence:

"Nine women can't make a baby in one month."

"Good news! I just hired three Java experts for your project! Please get them started right away – they're expensive, and we can't afford for them to be sitting around, OK? Oh, and by the way, how's the design coming along?"

"Well, if it will get us ahead we can hire another three teams and put them all on parallel component development and deliver earlier for more money. But it'll be worth it!"

Short Description:

It is critical to allocate the appropriate number of staff to the different phases of a project; schedule too many staff at the wrong time and the productivity level drops significantly; schedule too few staff and the deliveries are late. Delivery delays, cost overruns, and technical failure can result from not planning according to the chosen software development lifecycle. The overlap of software lifecycle phases can also have a significant impact on planned deliveries if staffing isn't increased accordingly.

Originally identified by Brooks in the *Mythical Man-Month* (1979), it involves the assumption that developers are interchangeable and that the number of people working on a project is inversely proportional to development. Brooks discusses how this assumption is false and presents what he calls Brooks' Law: "Adding manpower to a late project makes it later" [Brooks, 1979].

3.3.6 The Brawl

Anti-Pattern Name: The Brawl

Also Known As: The Anti-Patton

Re-Factored Solution Name: Leadership 101: Intro to Leadership Concepts

Re-Factored Solution Type: Role

Root Causes: Ignorance

Anecdotal Evidence:

"He's a great manager/engineer/programmer; he'll make a great project manager."

"This guy has got degrees from MIT and Harvard; he'll make a great project manager."

"We don't believe you have been taking enough risk with your project. We think you should change the architecture to accommodate the latest thinking in technology. This shouldn't have any impact on the rest of the project."

Short Description:

There are at least two distinct aspects to being a project manager: leadership and management. It is important that the two are understood and that effective leadership and management skills are applied to achieve success. A poor manager or leader typically results in project failure. However, leadership, more than management, can save a project. Often the battle cry of the engineer or the programmer is, "We need a leader, not a manager," or "There is no leadership." When this is the case, development teams muddle about, stagnate, and fail to attain their goals. This festering often results in a brawl where each faction of the development team attempts to position themselves into a leadership role.

Management cliques are very common and always negatively impact those around them because of the politics involved, causing a dysfunctional work environment. Management hierarchies are often based on personal relationships (so-called "old boys' clubs") and manager's favorite views of software development practices regardless of their management skill and experience.

3.3.7 The Domino Effect

Anti-Pattern Name: The Domino Effect

Also Known As:

Crap Rolls Downhill

Re-Factored Solution Name: Reverse Domino Tilting or Strategic Domino Removal

Re-Factored Solution Type: Process

Root Causes: Multiple

Anecdotal Evidence:

"Hey Rick, your project is doing great! Listen, we've decided to pull Ben from your team for just a couple weeks to help out on Jerry's project. I know he's your best Java guy, but it's only for a little while, and Jerry's project really needs some help. Besides, you're a little bit ahead of schedule, so pulling Ben may even help you out by giving some of your junior developers a chance to show some leadership! So talk to Ben about it, okay? He'll have to delay that vacation he's planned, too. They want him to start with Jerry this afternoon."

Short Description:

Project managers who treat the resources for each project for which they are responsible in a collective manner, blur project boundaries causing a domino effect of problems. The need to continually move the critical resources to cover gaps left by moving them from a previous project leads to software delivery delays, developer frustration, and lack of confidence in management, and in extreme cases, to developer attrition and project failure.

3.4 Analysis of Anti-Patterns in Laplante et al. (2005)

We will present in this section ten software project management anti-patterns identified by Laplante et al. (2005):

1. **Absentee Manager:** Manager who engages in avoidance behavior or is invisible for long periods of time.
2. **All You Have Is a Hammer:** One-dimensional management, where the same techniques are used on all subordinates and in all situations.
3. **Headless Chicken:** A very confused manager who lacks focus, has no plan, and never follows through with anything.
4. **Leader Not Manager:** Manager with vision but no plan or management methodology.
5. **Mushroom Management:** Isolating developers from end users, under the mistaken assumption that requirements are stable and well-understood by both end users and the software team at project inception.
6. **Proletariat Hero:** Assuming that coercion is an efficient way to increase productivity.
7. **Rising Upstart:** Superstars who cannot wait their time and want to skips learning phases.
8. **Road to Nowhere:** Lack of planning.
9. **Ultimate Weapon:** Relying heavily on a superstar in the team.
10. **Warm Bodies:** Assuming developers are interchangeable and that the number of people working on a problem is inversely proportional to development time.

3.4.1 Absentee Manager

Anti-Pattern Name: Absentee Manager

Anecdotal Evidence:

"You're there, he ain't."

Short Description:

Any manager who engages in avoidance behavior or is invisible for long periods of time – either hiding on premises or away from the office.

No one wants the constant intrusion of management, it indicates a lack of trust in the workforce, but there are obviously times when management must be visible because they are the primary decision makers. Whey key managers cannot be found, subordinates are left to make crucial decisions or those decisions are delayed. Either way, the manager's continued absence is hindering, or even damaging the company. Furthermore, it is demoralizing to employees when their boss is not putting in the time.

3.4.2 All You Have Is a Hammer

Anti-Pattern Name: All You Have Is a Hammer

Also Known As: One-Trick Pony

Anecdotal Evidence:

"If all you have is a hammer, everything is a nail."

Short Description:

One-dimensional management, where the same techniques are used on all subordinates and in all situations. In reality, if all you have is a hammer, everything is a thumb!

One of the biggest mistakes you can make is to impose your own values and win conditions on someone else. You might be motivated by money, or title, or whatever… someone else may not be, and assuming that everyone holds the same motivations, anxieties, and needs ensures that most of your subordinates are unhappy and unmotivated.

3.4.3 Headless Chicken

Anti-Pattern Name: Headless Chicken

Short Description:

A very confused manager who lacks focus, has no plan, and never follows through with anything.

Confused behavior; trying this idea and that; running from one problem to another, without focus. Those observing the Headless Chicken performance are themselves confused and demoralized.

3.4.4 Leader Not Manager

Anti-Pattern Name: Leader Not Manager

Short Description:

Being a great leader does not necessarily mean being a great manager. This anti-pattern illustrates the problem of having vision but no plan.

Inspiring leadership is very important, but so are the day-to-day operations of an organization, project, or team and to do them effectively requires management acumen that even great leaders can lack. Some might view these tasks as drudgery, but while leaders inspire, managers organize, plan, supervise, and advise.

3.4.5 Mushroom Management

Anti-Pattern Name: Mushroom Management

Short Description:

A situation in which management fails to communicate effectively with staff. Essentially, information is deliberately withheld to keep everyone "fat, dumb, and happy."

When members of the team do not really understand the big picture, the effects can be significant. It is somewhat insulting to assume that someone working on the front lines does not have a need to understand the bigger picture. Moreover, those who are working directly with customers, for example, might have excellent ideas that may have sweeping impact on the company. So, Mushroom Management can lead to low employee morale, turnover, missed opportunities, and general failure.

3.4.6 Proletariat Hero

Anti-Pattern Name: Proletariat Hero

Anecdotal Evidence:

"In my annual review, I only state targets I've already attained."

Short Description:

The "everyman" worker is held up as the ideal, when in reality, he is a prop being used to mask inadequacies of management. A form of labor discipline as a means of "motivating" staff that gives an excuse for management to raise output expectations… get more with less.

The basic, flawed assumption at work in the plight of the Proletariat Hero is that people are lazy. Managers who believe this then utilize some form of labor discipline that coerces workers to become more efficient. It assumes that people will work at the slowest rate that is not punished. It treats workers as unthinking cogs in a machine and is a mindset solely interested in efficiency. Paradoxically, removing individuality and creativity from work can often reduce productivity as people become bored and inefficient.

3.4.7 Rising Upstart

Anti-Pattern Name: Rising Upstart

Re-Factored Solution Name: Effective Mentoring

Short Description:

Rising upstarts are superstars that cannot wait their time and want to forego the requisite time to learn, mature, and find their place. This can sometimes be through ignorance (they do not know what they do not know), and sometimes it is through impatience (they know what others do not know). The Rising Upstart presents a real challenge to all but the most proficient managers.

Uncontrolled Rising Upstarts can cause an imbalance in an organization, a situation that can be difficult to manage. A happy equilibrium must be found that allows the cream to rise to the top in step with their talents (artificially holding them down will force these valuable assets to leave), but not too quickly that they prematurely crash and burn or offend the still effective old guard.

3.4.8 Road to Nowhere

Anti-Pattern Name: Road to Nowhere

Anecdotal Evidence:

"The plan is nothing, planning is everything." – Dwight D. Eisenhower

"When you don't know where you are going, all roads will take you there."

Short Description:

The lack of a plan causes confusion and a crisis of leadership.

Without a plan, how can you organize to achieve your goals? How can you properly partition the work that must be done and track its progress? How do you get buy-in from

your superiors, subordinates, and peers for the direction in which you are taking them? Furthermore, if you somehow manage to meet your goals, even by luck, how will you be able to repeat that success, learn from mistakes, and extract best processes? It is exceptionally difficult to make even the simplest of tactical decisions when a broader plan is not in place – it is like walking in the dark with your hand outstretched.

Even those managers who prefer *ad hoc* techniques and, through their hard work and luck, sometimes manage to be successful, the dissatisfaction and frustration of everyone else that has to work "blind" is palpable. Moreover, how are you going to learn from someone who does not even know why he succeeds himself?

3.4.9 Ultimate Weapon

Anti-Pattern Name: Ultimate Weapon

Short Description:

Phenoms can be relied upon so much by their peers or organization that they become the conduit for all things.

Ultimate Weapons can do great things, but often they know how great they are and their arrogance and self-righteousness can be just as great. If these individuals are not managed effectively, the harm they cause can overshadow their achievements.

Despite their talents, even a superstar cannot win on their own all the time, and moreover, the Ultimate Weapon's teammates can fell second-rate and forgotten.

Managers can become so seduced by the prodigious talent of these individuals that they involve them in everything. They become so important to their project, team, or even company, that without them success seems impossible.

3.4.10 Warm Bodies

Anti-Pattern Name: Warm Bodies

Short Description:

A management situation that involves the practice of moving workers on to late-running projects to help get them back on schedule. This is a flawed resource-leveling approach because it fails to account for time spent on the learning curve. New team members are therefore unproductive as they learn the ropes; but even worse, they also need assistance from the existing team members (thus reducing the productivity of the latter), and the work they do get done is often so error-riddled that it needs repair, thus reducing productivity even further.

In the software development world, Warm Bodies refer to those individuals who tend to be thrown at late projects under the assumption that increasing the resources will increase productivity. Brooks reported decades ago, however, that this assumption was flawed and that adding people to a late project in fact only makes it later [Brooks, 1979]. The best analogy Brooks makes is to pregnancy. It does not matter how many mothers you throw at it, it will still take nine months. Because there can be no breakdown of the tasks involved, and therefore no parallelism, the extra woman-power has no effect.

Obviously, software construction can be decomposed into more discrete tasks than pregnancy, but the general principle is still valid. If a developer is working on a device

driver, it will not get completed in a fifth of the time if you give him four buddies to help out. In fact, delays will result as he is forced to spend precious time explaining what he was doing, why he was doing it, and how it fits into the bigger picture. Consequently, utilizing Warm Bodies as a productivity driver is a complete fallacy.

3.5 Analysis of Anti-Patterns in c2.com (2014)

We will present in this section four software project management anti-patterns identified on the c2.com webpage (2014).

1. **An Athena:** Expecting staff to spring, fully formed, from whatever institution they attended, and thus, applying rigid selection criteria to prospective job applicants.
2. **Appointed Team:** False assumption that a group of people selected by management will immediately gel and become a team.
3. **Dry Waterhole:** Specifying stringent requirements for a job when it is not strictly necessary, resulting in a very limited pool of available talent.
4. **Glass Wall:** Only considering the candidate's past experience during staff selection.

3.5.1 An Athena

Anti-Pattern Name: An Athena

Short Description:

Staff is expected to spring, fully formed, from the brow of whatever institution they attended. Selection criteria are rigidly applied to prospective job applicants.

Over time, selection criteria become standard and, like the goddess Athena, successful candidates are expected to enter a new job fully formed by their previous experiences.

When it occurs on a large scale, this practice locks people into roles they have already performed. The opportunity to move on and learn new skills is greatly limited; all learning is presumed to have occurred previously. Taken to its logical conclusion, this means college.

3.5.2 Appointed Team

Anti-Pattern Name: Appointed Team

Re-Factored Solution Name: Self-Selecting Team

Short Description:

The false assumption that a group of people selected by management will immediately gel and become a team. There are no perfect criteria for screening team members. However, when teams are appointed according to management insight, needs, and biases, it often results in disempowered teams unwilling to take extraordinary measures to meet project goals.

3.5.3 Dry Waterhole

Anti-Pattern Name: Dry Waterhole

Short Description:

You get into the habit of specifying stringent requirements for a job when it is not strictly necessary. Over time this habit spreads to other employers, and the pool of available talent dries up as lesser experienced people are denied the opportunity to get experience.

3.5.4 Glass Wall

Anti-Pattern Name: Glass Wall

Anecdotal Evidence:

"You are only as good as you were!"

Short Description:

Prevents people from rising as far in their profession as others or expanding into other fields, not because they lack the ability, but because they lack the relevant experience.

When hiring someone, an obvious starting point is their resume. This naturally focuses on what someone has done in the past. However, what that individual wants to achieve in the future is not covered, or is briefly touched on during the interview. Thus, candidates are filtered out and selected purely on the basis of what they have done.

4 Consolidated Anti-Pattern List

4.1 Introduction

As the reader may have noticed, some of the anti-patterns analyzed in the previous chapter are either repeated or have different names for a similar anti-pattern manifestation. Therefore, in this chapter we present a consolidated list of anti-patterns, which can be more easily managed.

4.2 Consolidated List

The consolidated list is shown in Table 1 and has six columns. The first column is simply the anti-pattern name. Columns two through five are the different names used by each author for the given anti-pattern. The last column shows the main reference that was chosen to extract information about the given anti-pattern.

The reference shown in the last column of Table 1 also indicates where the anti-pattern name came from. The only exception to the rule is the *Chaos* anti-pattern, which has been renamed here to *Inflexible Plan* in order to better represent the malpractices behind the anti-pattern and thus avoid confusion.

We have also consolidated anti-patterns that we believe to be very similar in their description, context, and solution. Four anti-pattern consolidations were made in this work:

- **An Athena:** Merged with *Dry Waterhole* as both discuss the problem of unnecessarily restricting the profile of candidates considered in the staff selection process.
- **Glass Wall:** Merged with *Dry Waterhole* as both discuss the problem of unnecessarily restricting the profile of candidates considered in the staff selection process.
- **Headless Chicken:** Merged with *Irrational Management* as both discuss the problems caused by a confused manager, characterized by indecisiveness and lack of focus.
- **Warm Bodies:** Merged with *Size Isn't Everything* as both discuss the mistaken assumption that developers are interchangeable and that the number of people working on a problem is inversely proportional to the time it takes to solve it.

Most of the anti-patterns in the consolidated list in Table 1 come from the work of Brown et al. Their work was later extended by Laplante et al, who considered two anti-patterns identified by Brown et al., *Irrational Management* and *Project Mismanagement*, to be a combination of several, more precise, anti-patterns.

Even though the work by Laplante et al. thoroughly expands these two anti-patterns, we have decided to include both *Irrational Management* and *Project Mismanagement* in our consolidated list of anti-patterns. We believe further anti-patterns could be identified within the wide contexts of the *Irrational Management* and the *Project Mismanagement* anti-patterns.

Anti-Pattern Name	Brown et al., 1998	Brown et al., 2000	Laplante et al., 2005	c2.com	Final Reference
Absentee Manager	-	-	Absentee Manager	-	Laplante et al., 2005
All You Have Is a Hammer	-	-	All You Have Is a Hammer	-	Laplante et al., 2005
Appointed Team	-	-	-	Appointed Team	c2.com
Detailitis Plan	Detailitis Plan	-	-	-	Brown et al., 1998
Dry Waterhole	-	-	-	An Athena, Dry Waterhole, Glass Wall	c2.com
Fire Drill	Fire Drill	-	-	-	Brown et al., 1998
Glass Case Plan	Glass Case Plan	-	-	-	Brown et al., 1998
Inflexible Plan	-	Chaos	-	-	Brown et al., 2000
Irrational Management	Irrational Management	-	Headless Chicken	-	Brown et al., 1998
Leader Not Manager	-	-	Leader Not Manager	-	Laplante et al., 2005
Micro-Management	-	Micro-Management	-	-	Brown et al., 2000
Mushroom Management	Mushroom Management	-	Mushroom Management	-	Brown et al., 1998
Myopic Delivery	-	Myopic Delivery	-	-	Brown et al., 2000
Process Disintegration	-	Process Disintegration	-	-	Brown et al., 2000
Project Mismanagement	Project Mismanagement	-	-	-	Brown et al., 1998
Proletariat Hero	-	-	Proletariat Hero	-	Laplante et al., 2005
Rising Upstart	-	-	Rising Upstart	-	Laplante et al., 2005
Road to Nowhere	-	-	Road to Nowhere	-	Laplante et al., 2005
Size Isn't Everything	Warm Bodies	Size Isn't Everything	Warm Bodies	-	Brown et al., 2000
The Brawl	-	The Brawl	-	-	Brown et al., 2000
The Domino Effect	-	The Domino Effect	-	-	Brown et al., 2000
Ultimate Weapon	-	-	Ultimate Weapon	-	Laplante et al., 2005

Table 1: Consolidated list of software project management anti-patterns.

We have included a second, more simplified table of anti-patterns, shown in Table 2, to be used as a quick reference or lookup table. It has three columns; column one contains the anti-pattern name, column two contains a brief description of the anti-pattern, and column three contains the references where the anti-pattern comes from, with the main reference shown in bold type.

Anti-Pattern Name	**Description**	**References**
Absentee Manager	Manager who engages in avoidance behavior or is invisible for long periods of time.	**Laplante et al., 2005**
All You Have Is a Hammer	One-dimensional management, where the same techniques are used on all subordinates and in all situations.	**Laplante et al., 2005**
Appointed Team	False assumption that a group of people selected by management will immediately gel and become a team.	**c2.com 2014**
Detailitis Plan	Excessive planning leading to complex schedules with high level of detail, giving the false perception that the project is fully under control.	**Brown et al., 1998**
Dry Waterhole	Specifying stringent requirements for a job when it is not strictly necessary, resulting in a very limited pool of available talent.	**c2.com 2014**
Fire Drill	Months of boredom followed by demands for immediate delivery.	**Brown et al., 1998**
Glass Case Plan	Lack of tracking and updating of initial plans, assuming the plan is enough.	**Brown et al., 1998**
Inflexible Plan	Lack of flexible plans and processes.	**Brown et al., 2000**
Irrational Management	Irrational management decisions, habitual indecisiveness and other negative management practices.	**Brown et al., 1998**, Laplante et al., 2005
Leader Not Manager	Manager with a vision (leader) but no plan or management methodology.	**Laplante et al., 2005**
Micro-Management	Excessive management involvement in tasks beyond their responsibility.	**Brown et al., 2000**
Mushroom Management	Isolating developers from end users, under the mistaken assumption that requirements are stable and well-understood by both end users and the software team at project inception.	**Brown et al., 1998**, Laplante et al., 2005
Myopic Delivery	Management insisting on original delivery date even when reducing staff or funding.	**Brown et al., 2000**
Process Disintegration	Failing processes due to an underlying decline in overall cooperation and morale.	**Brown et al., 2000**
Project Mismanagement	Lack of proper software project monitoring and controlling.	**Brown et al., 1998**
Proletariat Hero	False assumption that coercion is an efficient way to increase productivity.	**Laplante et al., 2005**
Rising Upstart	Superstars who cannot wait their time and want to skip learning phases.	**Laplante et al., 2005**
Road to Nowhere	Lack of planning.	**Laplante et al., 2005**
Size Isn't Everything	Assuming developers are interchangeable and that the number of people working on a problem is inversely proportional to development time.	Brown et al., 1998, **Brown et al., 2000**, Laplante et al., 2005
The Brawl	Project manager with no leadership or management experience.	**Brown et al., 2000**
The Domino Effect	Moving critical resources between projects, blurring project boundaries.	**Brown et al., 2000**
Ultimate Weapon	Relying heavily on a superstar in the team.	**Laplante et al., 2005**

Table 2: Description of software project management anti-patterns.

5 Anti-Pattern Categorization

5.1 Introduction

In this chapter we will categorize the anti-patterns in our consolidated list. We will first explain the criteria we chose to categorize anti-patterns, and then present the results of our anti-pattern categorization. We will end the chapter with an explanation of an anti-pattern categorization tool developed in this thesis.

5.2 Criteria

In order to categorize the consolidated list of anti-patterns, we will use four criteria:

1. The software project management activity (or activities) impacted by the anti-pattern.
2. The role(s) in the organization most impacted by the anti-pattern.
3. The root cause(s) of the anti-pattern.
4. The anti-pattern solution type(s).

We initially considered the inclusion of development phase as one of the criteria, but upon a preliminary categorization, it became clear that most of the anti-patterns would impact all phases of development. We thus decided to remove development phase from the list of criteria.

5.2.1 Software Project Management Activities

According to Kerzner (2013), there are five main activities in software project management: planning, scheduling, controlling, staffing, and motivating. These are explained in further detail next.

- **Planning:** A proactive activity, which involves creating a project plan that includes, as a minimum, a business case, a cost/benefit analysis, a risk analysis, and task delineation, including estimated resources and schedule [Peters, 2008].
- **Scheduling:** The creation of the project schedule, which can be described as a list of events where each event has a calendar date for starting and finishing. The schedule includes resource and time allocation for each task [Peters, 2008]. Scheduling and planning, although different, are largely related activities. We have decided to follow Kerzner (2013) and Peters (2008) in this work and keep the two activities separated.
- **Staffing:** Consists of recruiting the correct people while keeping teams small, reviewing and evaluating team effectiveness and productivity and removing roadblocks when needed, as well as being aware of team conduct and disruption [Peters, 2008].
- **Controlling:** An activity which is both passive, monitoring using Earned Value Management, and active, re-planning to respond to some unexpected event, such as tasks running behind schedule, over budget, or changing requirements [Peters, 2008].

- **Motivating:** Creating and maintaining a collegial environment where teams participate in the creation of their schedules and goals and are efficiently led and managed by a "servant leader", who relies on his/her strengths, selects a complimentary team, and negotiates with teams, other leaders, and managers to achieve challenges [Peters, 2008].

5.2.2 Impacted Roles

Brown et al. present anti-patterns from three main viewpoints: the software developer, the software architect, and the software manager. For the purpose of this thesis, we have consolidated the architect role into the developer role as both are often subordinates of the project manager.

We initially considered including the customer role and the upper management role in our analysis; two roles that, based on our own experience, we believe are also important in any organization. However, upon a preliminary categorization, it became clear that the upper management role would be significantly less impacted, as compared to other roles, by the software project management anti-patterns in our consolidated list. We therefore decided to remove the role of upper management from our analysis.

We have thus identified three major roles that would be impacted by software project management anti-patterns: the developer, the manager, and the customer. These three roles are explained in further detail next.

- **Developer:** The developer role includes not only software developers, but also any member of the development team, such as analysts, architects, designers, testers, team leaders, and support personnel. This role directly or indirectly reports to the manager role.
- **Manager:** The manager role is viewed here as the project manager; the person in charge of the project and responsible for managing and leading the development team to success, to whom the development team reports to.
- **Customer:** The customer role is the person(s), team, or organization, which will ultimately use the product created by the development team. According to McConnell (1998), the customer is “the person or persons who the software ultimately must please in order to be considered a success.” This role can be internal to the organization, such as teams that rely on artifacts generated by other teams, or external, such as a business client.

5.2.3 Root Causes

As discussed earlier in section 2.1.4, Brown et al. (1998) have identified eight main root causes in software development mistakes or malpractices: haste, apathy, narrow-mindedness, sloth, avarice, ignorance, pride, and responsibility. For the purpose of this work, we decided to consolidate *apathy* into *sloth* and *narrow-mindedness* into *ignorance* as we believe they are similar enough, making it hard to identify their boundaries. We have also decided not to use the root cause *responsibility* as Brown et al. do not provide a complete definition for it.

We will thus use five main root causes in this work: avarice, haste, ignorance, pride, and sloth.

- **Avarice:** "Greed can take many forms, but it leads to inappropriate software development decisions" [Brown et al., 1998]. Our interpretation of greed in this work is broader than the pure monetary aspect; we will use avarice to describe ambitious behaviors not only related to money, but also people, project schedules, and delivery dates, for example.
- **Haste:** "Hasty decisions lead to compromises in software quality. Software projects are often subjected to severe schedule-related stress. At project inception, managers are pressured to trim budgets and schedules to make unrealistic targets. As successive project deadlines are missed, anything that appears to work is considered acceptable, regardless of quality" [Brown et al., 1998].
- **Ignorance:** "Ignorance is intellectual sloth. It's the result of failing to seek understanding. It keeps people stupid, and eventually leads to long-term software problems" [Brown et al., 1998].
- **Pride:** "The sin of pride is the not-invented-here syndrome" [Brown et al., 1998].
- **Sloth:** "Sloth is the "healthy sign" of a lazy developer or manager, who makes poor decisions based upon an "easy answer" [Brown et al., 1998]. We also interpret sloth as the reluctance of management to make tough decisions, which could include, for example, a situation where the project manager is unwilling to allocate necessary resources to a task or project.

5.2.4 Solution Types

According to Brown et al. (1998), there are four types of solutions to address anti-patterns: process, role, software, technology. They identify the type of action that results from the anti-pattern solution. We have decided not to use the software solution type as we have found it to be not relevant to software project management anti-patterns. In other words, none of the anti-patterns in our consolidated list can be effectively addressed through the development of new software. However, we have included a new solution type called *training*, which is an area not covered by any of the solution types proposed by Brown et al.

We will thus use four main solution types in this work: process, role, technology, and training.

- **Process:** "Indicates that the solution entails pursuing a process. Process patterns provide the definition of activities that are consistently repeatable for a given solution" [Brown et al., 1998].
- **Role:** "Indicates that the solution entails assigning responsibility to an individual or group. Role patterns solve problems by allocating clear responsibilities to organizational stakeholders" [Brown et al., 1998].
- **Technology:** "Technology indicates that the solution entails acquisition of a technology or product. Technology patterns solve software problems through adoption of a technology, as opposed to programming the capability from scratch" [Brown et al., 1998].

- **Training:** "A process by which someone is taught the skills that are needed for an art, profession, or job" [Merriam-Webster, 2014]. In this work we consider such process to include any professional development activity, such as education and on-the-job training.

5.3 Results

In this section we will present and analyze the results of our anti-pattern categorization using the four criteria explained in section 5.2 – impacted software project management activities, most impacted roles, root causes, and solution types.

It is worth pointing out that some anti-patterns may impact more than one software project management activity or role, or may have more than one root cause or solution type. In this work we have attempted to find the single most important factor for each criteria; e.g. the most significant root cause. In most cases we were able to reduce it to a single factor, but there are cases with multiple major factors.

Please also note that the categorization activity does not have a single solution and many alternative results can also be justified. By no means do we attempt for completeness in our categorization analysis; the intent and scope of this work is simply to highlight the major factors related to each software project management anti-pattern.

5.3.1 Categorization per Software Project Management Activity

None of the anti-patterns in the consolidated list had the impacted software project management activity identified by the author. We performed this analysis looking for the single most impacted activity among the five activities described in section 5.2.1, and based on the anti-pattern definition and characteristics, presented earlier in Chapter 3.

This categorization has been especially difficult, as most anti-patterns affect most of the activities in project management, but we have tried to choose the most representative among the impacted activities.

The final categorization is shown in Table 3. Note that while we have strived to find a single most impacted activity, such as in *All You Have Is a Hammer* which mostly impacts *Motivating*, there are cases in which anti-patterns may have a major impact on more than one activity, such as in *Absentee Manager* which significantly impacts both *Controlling* and *Motivating*.

Also note that the table has been sorted by the criteria, in this case the impacted activity, not the anti-pattern name.

Anti-Pattern Name	**Impacted Software Project Management Activity**
Fire Drill	Controlling
Glass Case Plan	Controlling
Mushroom Management	Controlling
Myopic Delivery	Controlling
Project Mismanagement	Controlling
The Domino Effect	Controlling
Absentee Manager	Controlling, Motivating
Irrational Management	Controlling, Motivating
The Brawl	Controlling, Motivating
All You Have Is a Hammer	Motivating
Micro-Management	Motivating
Process Disintegration	Motivating
Proletariat Hero	Motivating
Rising Upstart	Motivating
Ultimate Weapon	Motivating
Inflexible Plan	Planning
Leader Not Manager	Planning
Road to Nowhere	Planning
Detailitis Plan	Scheduling
Size Isn't Everything	Scheduling
Appointed Team	Staffing
Dry Waterhole	Staffing

Table 3: Software project management anti-patterns categorized by impacted activity.

5.3.1.1 Planning

There are three anti-patterns whose most impacted software project management activity is planning:

- **Inflexible Plan:** "The impact of change on the project usually affects one or more of three strategic project facets: schedule, cost, and quality" [Brown et al., 2000].
- **Leader Not Manager:** "Inspiring leadership is very important, but so are the day-to-day operations of an organization, project, or team, such as budgeting, organizational management, and planning" [Laplante et al., 2005].
- **Road to Nowhere:** This anti-pattern is characterized by the lack of a plan [Laplante et al., 2005].

It means that approximately 14% of the anti-patterns in the consolidated list have planning as the most impacted software project management.

5.3.1.2 Scheduling

There are two anti-patterns whose most impacted software project management activity is scheduling:

- **Detailitis Plan:** "The objective shifts from delivery of software to delivery of a set of plans. Management mistakenly assumes that because effort and cost are tracked, progress must be equivalent" [Brown et al., 1998].
- **Size Isn't Everything:** The false assumption that developers are interchangeable and that the number of people working on a project is inversely proportional to development time often results in overstaffing or understaffing [Brown et al., 2000].

It means that approximately 9% of the anti-patterns in the consolidated list have scheduling as the most impacted software project management, making it one of the two least impacted activities in our analysis.

5.3.1.3 Staffing

There are two anti-patterns whose most impacted software project management activity is staffing:

- **Appointed Team:** Appointing teams based on management insight, needs, and biases results in "a disempowered team unwilling to take extraordinary measures to meet project goals" [c2.com, 2014].
- **Dry Waterhole:** While it is certainly essential to hire the correct people, being excessively stringent about the job requirements or the candidate profile may result in a very limited pool of available talent [c2.com, 2014].

It means that approximately 9% of the anti-patterns in the consolidated list have staffing as the most impacted software project management, making it one of the two least impacted activities in our analysis.

5.3.1.4 Controlling

There are nine anti-patterns whose most impacted software project management activity is controlling:

- **Absentee Manager:** Controlling is one of the two activities most impacted by an absent manager. A manager who is not present cannot effectively control a project.
- **Fire Drill:** A Fire Drill scenario starts when management decides that development must progress immediately, often making unrealistic demands for software delivery. "Because the entire project is pressed for time, compromises are willingly made in software quality and testing" [Brown et al., 1998].
- **Glass Case Plan:** The false assumption that the project plan guarantees project success results in management not knowing the status of the project's development; "the plan has no meaning, and control of delivery lessens as time goes on" [Brown et al., 1998].
- **Irrational Management:** Controlling is one of the two activities most impacted by irrational management. This anti-pattern is characterized by irrational management

decisions, indecisiveness, among other negative management habits that directly affect monitoring and controlling of a software project.

- **Mushroom Management:** There is a flawed assumption that requirements are stable and well understood by both end users and the software team at project inception; developers are thus kept isolated from end users by management resulting in poor design decisions, often with weak interfaces that do not fulfill the functional requirements [Brown et al., 1998].
- **Myopic Delivery:** Management's refusal to accept changes to the original schedule, even when reducing staff or funding results, among other things, in missed delivery date, limited functionality, poor reliability, or even project cancellation [Brown et al., 2000].
- **Project Mismanagement:** This anti-pattern is characterized by a lack of proper monitoring and controlling [Brown et al., 1998].
- **The Brawl:** Controlling is one of the two activities most impacted by this anti-pattern. "A project manager with no leadership or management experience and knowledge will typically struggle when some type of challenge is presented to them in which adjustments are difficult. They are unable to adapt, overcome, and succeed just because they haven't been there before" [Brown et al., 2000].
- **The Domino Effect:** "Treating resources for different projects in a collective manner, blurring project boundaries will likely result in no single project staying on schedule or within budget, and a general state of crisis management for all projects" [Brown et al., 2000].

It means that approximately 41% of the anti-patterns in the consolidated list have controlling as the most impacted software project management, making it one of the two most impacted activities in our analysis.

5.3.1.5 Motivating

There are nine anti-patterns whose most impacted software project management activity is motivating:

- **Absentee Manager:** Motivating is one of the two activities most impacted by an absent manager. "It is demoralizing to employees when their boss is not putting in the time" [Laplante et al., 2005].
- **All You Have Is a Hammer:** "Assuming that everyone holds the same motivations, anxieties, and needs ensures that most of your subordinates are unhappy and unmotivated" [Laplante et al., 2005].
- **Irrational Management:** Motivating is one of the two activities most impacted by irrational management. "A manager's inability to direct the development staff results in increased staff frustration" [Brown et al., 1998]; the staff itself becomes confused and demoralized [Laplante et al., 2005].
- **Micro-Management:** This anti-pattern often begins with a lack of knowledge about people management by the project manager; the micro-management behavior then becomes an attempt in mitigating the manager's weaknesses, and is characterized by over management of certain project aspects, aspects that are under the responsibility

of their subordinates. The range of micro-management can vary from an occasional fire drill to managing daily developer tasks. The most significant consequence of micro-management is developer frustration, often leading to loss of staff [Brown et al., 2000].

- **Process Disintegration:** "Despite its name this anti-pattern is primarily a people-caused anti-pattern that has process consequences" [Brown et al., 2000]. When processes begin to fail, the problem is often not with the process itself, but with the attitude of the people performing them. It is thus necessary to improve the morale of the development team.
- **Proletariat Hero:** There is a mistaken assumption by management that coercion is an efficient way to increase productivity and that treating people like machines will yield benefits. "Paradoxically, removing individuality and creativity from work can often reduce productivity as people become bored and inefficient" [Laplante et al., 2005].
- **Rising Upstart:** "Rising Upstarts are superstars that cannot wait their time and want to forego the requisite time to learn, mature, and find their place. Uncontrolled Rising Upstarts can cause an imbalance in an organization, a situation that can be difficult to manage. Equilibrium must be found that allows Upstarts to rise to the top in step with their talents – artificially holding them down will force these valuable assets to leave – but not too quickly that they prematurely crash and burn or they offend the still effective old guard" [Laplante et al., 2005].
- **The Brawl:** Motivating is one of the two activities most impacted by this anti-pattern. A project manager with no leadership or management experience and knowledge will often result in staff attrition and a decline in morale [Brown et al., 2000].
- **Ultimate Weapon:** Ultimate Weapons are extremely talented individuals in a software team, who, according to Laplante et al. (2005), "can do great things, but often they know how great they are and their arrogance and self-righteousness can be just as great. If these individuals are not managed effectively, the harm they cause can overshadow their achievements; teammates can feel second-rate and forgotten, and standardized processes and practices can be skipped."

It means that approximately 41% of the anti-patterns in the consolidated list have motivating as the most impacted software project management, making it one of the two most impacted activities in our analysis.

5.3.1.6 Summary

A summary of these numbers is shown in Table 4; the two most impacted software project management activities being *Controlling* and *Motivating*, and the two least impacted activities being *Scheduling* and *Staffing*. Note that given how some anti-patterns impact more than one activity, the sum of the percentages will not be 100%.

Table 5 further dissects the impact on the activities of *Controlling* and *Motivating*, showing that a total of 15 out of 22 anti-patterns, or 68%, impact at least one of these two activities.

Impacted Software Project Management Activity	**Number of Anti-Patterns**	**Percentage (out of 22 Anti-Patterns)**
Controlling	9	41%
Motivating	9	41%
Planning	3	14%
Scheduling	2	9%
Staffing	2	9%

Table 4: Activities most impacted by software project management anti-patterns.

Impacted Software Project Management Activity	**Number of Anti-Patterns**	**Percentage (out of 22 Anti-Patterns)**
Controlling-only	6	27%
Motivating-only	6	27%
Both Controlling *and* Motivating	3	14%
Total for Controlling *and/or* Motivating	15	68%

Table 5: Analysis of impact on *controlling* and *motivating* activities.

5.3.2 Categorization per Impacted Role

None of the anti-patterns in the consolidated list had the impacted role identified by the author. We performed this analysis looking for the single most impacted role among the four roles described in section 5.2.2, and based on the anti-pattern definition and characteristics, presented earlier in Chapter 3.

The final categorization is shown in Table 6. Note that while we have strived to find a single most impacted role, such as in *Micro-Management* which mostly impacts developers, there are cases in which anti-patterns may have a major impact on more than one role, such as in *Myopic Delivery* which significantly impacts both customers and developers.

Also note that, as in the previous section, the table has been sorted by the criteria, in this case the impacted role, not the anti-pattern name.

Anti-Pattern Name	**Impacted Role**
Fire Drill	Customer
Mushroom Management	Customer
Myopic Delivery	Customer, Developer
Project Mismanagement	Customer, Manager
Absentee Manager	Developer
All You Have Is a Hammer	Developer
Appointed Team	Developer
Irrational Management	Developer
Leader Not Manager	Developer
Micro-Management	Developer
Proletariat Hero	Developer
Road to Nowhere	Developer
The Brawl	Developer, Manager
The Domino Effect	Developer, Manager
Detailitis Plan	Manager
Dry Waterhole	Manager
Glass Case Plan	Manager
Inflexible Plan	Manager
Process Disintegration	Manager
Rising Upstart	Manager
Size Isn't Everything	Manager
Ultimate Weapon	Manager

Table 6: Software project management anti-patterns categorized by impacted role.

5.3.2.1 Developer

There are eleven anti-patterns whose most impacted role is the developer:

- **Absentee Manager:** "There are times when management must be visible because they are the primary decision makers. When key managers cannot be found, subordinates are left to make crucial decisions or those decisions are delayed.

Furthermore, it is demoralizing to employees when their boss is not putting in the time" [Laplante et al., 2005].

- **All You Have Is a Hammer:** "Assuming that everyone holds the same motivations, anxieties, and needs ensures that most of your subordinates are unhappy and unmotivated" [Laplante et al., 2005].
- **Appointed Team:** Appointing teams based on management insight, needs, and biases results in "a disempowered team unwilling to take extraordinary measures to meet project goals" [c2.com, 2014].
- **Irrational Management:** "A manager's inability to direct the development staff results in increased staff" frustration [Brown et al., 1998]; the staff itself becomes confused and demoralized [Laplante et al., 2005].
- **Leader Not Manager:** "Inspiring leadership is very important, but so are the day-to-day operations of an organization, project, or team and to do them effectively requires management acumen that even great leaders can lack" [Laplante et al., 2005]. An absence of effective management results, among other things, in staffing problems and unfocused workers.
- **Micro-Management:** This anti-pattern often begins with a lack of knowledge about people management by the project manager; the micro-management behavior then becomes an attempt in mitigating the manager's weaknesses, and is characterized by over management of certain project aspects, aspects that are under the responsibility of their subordinates. The range of micro-management can vary from an occasional fire drill to managing daily developer tasks. The most significant consequence of micro-management is developer frustration, often leading to loss of staff [Brown et al., 2000].
- **Myopic Delivery:** Developer is one of the two roles most impacted by this anti-pattern. Management's refusal to accept changes to the original schedule, even when reducing staff or funding results, among other things, in low morale, high attrition and stress [Brown et al., 2000].
- **Proletariat Hero:** There is a mistaken assumption by management that coercion is an efficient way to increase productivity and that treating people like machines will yield benefits. "Paradoxically, removing individuality and creativity from work can often reduce productivity as people become bored and inefficient" [Laplante et al., 2005].
- **Road to Nowhere:** The lack of a plan causes confusion and a crisis of leadership, where people who have to work with the "blind" manager quickly become dissatisfied and frustrated [Laplante et al., 2005].
- **The Brawl:** Developer is one of the two roles most impacted by this anti-pattern. A project manager with no leadership or management experience and knowledge will often result in staff attrition and a decline in morale [Brown et al., 2000].
- **The Domino Effect:** Developer is one of the two roles most impacted by this anti-pattern. "Treating resources for different projects in a collective manner, blurring project boundaries will likely result in retraction of developers into defensive posture that eliminates collaboration and sharing, frequent personnel crises, and high personnel turnover" [Brown et al., 2000].

It means that 50% of the anti-patterns in the consolidated list have the developer as the most impacted role, making it one of the two most impacted roles in our analysis.

5.3.2.2 Manager

There are eleven anti-patterns whose most impacted role is the manager:

- **Detailitis Plan:** "The objective shifts from delivery of software to delivery of a set of plans. Management mistakenly assumes that because effort and cost are tracked, progress must be equivalent" [Brown et al., 1998].
- **Dry Waterhole:** While it is certainly essential to hire the correct people, being excessively stringent about the job requirements or the candidate profile may result in a very limited pool of available talent [c2.com, 2014].
- **Glass Case Plan:** The false assumption that the project plan guarantees project success results in management not knowing the status of the project's development; "the plan has no meaning, and control of delivery lessens as time goes on" [Brown et al., 1998].
- **Inflexible Plan:** "The impact of change on the project usually affects one or more of three strategic project facets: schedule, cost, and quality" [Brown et al., 2000]; three facets under direct responsibility of the project manager.
- **Process Disintegration:** When processes begin to fail, a common consequence is a lack of awareness of the true state of project activities and deliverables [Brown et al., 2000].
- **Project Mismanagement:** Manager is one of the two roles most impacted by project mismanagement. A software project that lacks proper monitoring and controlling may result in cost overruns or even premature termination of the project [Brown et al., 1998].
- **Rising Upstart:** "Rising Upstarts are superstars that cannot wait their time and want to forego the requisite time to learn, mature, and find their place. They present a real challenge to all but the most proficient managers and require close management" [Laplante et al., 2005].
- **Size Isn't Everything:** The false assumption that developers are interchangeable and that the number of people working on a project is inversely proportional to development time often results in overstaffing or understaffing. In either scenario, management is likely to be confronted with incremental delivery delays, cost overruns, and eventual technical failure [Brown et al., 2000].
- **The Brawl:** Manager is one of the two roles most impacted by this anti-pattern. A project manager with no leadership or management experience and knowledge will typically struggle when some type of challenge is presented to them in which adjustments are difficult. They are unable to adapt, overcome, and succeed just because they haven't been there before [Brown et al., 2000].
- **The Domino Effect:** Manager is one of the two roles most impacted by this anti-pattern. "Treating resources for different projects in a collective manner, blurring project boundaries will likely result in no single project staying on schedule or within budget, and a general state of crisis management for all projects" [Brown et al., 2000].

- **Ultimate Weapon:** Ultimate Weapons are extremely talented individuals in a software team, who, according to Laplante et al. (2005), "can do great things, but often they know how great they are and their arrogance and self-righteousness can be just as great. If these individuals are not managed effectively, the harm they cause can overshadow their achievements; teammates can feel second-rate and forgotten, and standardized processes and practices can be skipped".

It means that 50% of the anti-patterns in the consolidated list have the manager as the most impacted role, making it one of the two most impacted roles in our analysis.

5.3.2.3 Customer

There are four anti-patterns whose most impacted role is the customer:

- **Fire Drill:** A Fire Drill scenario starts when management decides that development must progress immediately, often making unrealistic demands for software delivery. "Because the entire project is pressed for time, compromises are willingly made in software quality and testing" [Brown et al., 1998].
- **Mushroom Management:** There is a flawed assumption that requirements are stable and well understood by both end users and the software team at project inception; developers are thus kept isolated from end users by management resulting in poor design decisions, often with weak interfaces that do not fulfill the functional requirements [Brown et al., 1998].
- **Myopic Delivery:** Customer is one of the two roles most impacted by this anti-pattern. Management's refusal to accept changes to the original schedule, even when reducing staff or funding may result, among other things, in missed delivery date, limited functionality, poor reliability, or even project cancellation [Brown et al., 2000].
- **Project Mismanagement:** Customer is one of the two roles most impacted by project mismanagement. A software project that lacks proper monitoring and controlling may result in the development of the wrong product, with missed or incorrect functionality. [Brown et al., 1998].

It means that approximately 18% of the anti-patterns in the consolidated list have the customer as the most impacted role, making it the least impacted role in our analysis.

5.3.2.4 Summary

A summary of these numbers is shown in Table 7; the least impacted role being the customer and the two most impacted roles being the developer and the manager. Note that given how some anti-patterns impact more than one role, the sum of the percentages will not be 100%.

Table 8 further dissects the impact on the roles of the developer and the manager, showing that a total of 20 out of 22 anti-patterns, or 91%, impact at least one of these two roles.

It is interesting to note that 50% of the anti-patterns in our consolidated list are suffered by the role of the manager, who in many cases might be causing his own demise. In the next section we will shed more light into this topic as we look into anti-patterns root causes

Impacted Role	**Number of Anti-Patterns**	**Percentage (out of 22 Anti-Patterns)**
Developer	11	50%
Manager	11	50%
Customer	4	18%

Table 7: Roles most impacted by software project management anti-patterns.

Impacted Role	**Number of Anti-Patterns**	**Percentage (out of 22 Anti-Patterns)**
Developer-only	9	41%
Manager-only	9	41%
Both Developer *and* Manager	2	9%
Total for Developer *and/or* Manager	20	91%

Table 8: Analysis of impact on *developer* and *manager* roles.

5.3.3 Categorization per Root Cause

Some of the anti-patterns in the consolidated list, in particular those that came from Brown, 1998, already had their root causes identified by the author. However, we replicated this analysis looking for the single major root cause among the five causes described in section 5.2.3. For those anti-patterns that did not have this information, we identified their main root causes based on the anti-pattern definition and characteristics, presented earlier in Chapter 3.

In order to validate our results, we inverted the question of "what is the root cause for anti-pattern X?" into "if management displays root cause Y type of behavior, what are the possible problem scenarios that might occur?"

The final categorization is shown in Table 9. Note that while we have strived to find a single major root cause for these anti-patterns, such as *The Brawl* which is rooted in ignorance, there are cases in which anti-patterns may have more than one major root cause, such as *Project Mismanagement* which is rooted in ignorance, and sloth.

Also note that, as in previous sections, the table has been sorted by the criteria, in this case the root cause, not the anti-pattern name.

Anti-Pattern Name	**Root Cause**
Detailitis Plan	Avarice
Dry Waterhole	Avarice
Fire Drill	Haste
Irrational Management	Ignorance
All You Have Is a Hammer	Ignorance
Appointed Team	Ignorance
Glass Case Plan	Ignorance
Leader Not Manager	Ignorance
Mushroom Management	Ignorance
Proletariat Hero	Ignorance
The Brawl	Ignorance
Rising Upstart	Ignorance
Size Isn't Everything	Ignorance
The Domino Effect	Ignorance
Micro-Management	Ignorance, Pride
Inflexible Plan	Ignorance, Sloth
Project Mismanagement	Ignorance, Sloth
Myopic Delivery	Pride
Ultimate Weapon	Pride, Sloth
Absentee Manager	Sloth
Process Disintegration	Sloth
Road to Nowhere	Sloth

Table 9: Software project management anti-patterns categorized by root cause.

5.3.3.1 Avarice

There are two anti-patterns rooted in avarice:

- **Detailitis Plan:** The excessive planning involved in this anti-pattern is due to the ambition of capturing all details and tasks of a project.
- **Dry Waterhole:** Staff selection is a vital process in any organization, but it can be severely hampered by the greed of looking for a very specific profile in candidates.

It means that approximately 9% of the anti-patterns in the consolidated list are due to avarice on the part of the project manager.

5.3.3.2 Haste

There is one anti-pattern rooted in haste:

- **Fire Drill:** A Fire Drill starts when management decides that development must progress immediately, often making unrealistic demands for software delivery. It is this haste to deliver the product that prevents management from considering a properly revised schedule.

It means that approximately 5% of the anti-patterns in the consolidated list are due to haste on the part of the project manager, making it the least common root cause in our analysis.

5.3.3.3 Ignorance

There are fourteen anti-patterns rooted in ignorance:

- **All You Have Is a Hammer:** One-dimensional management is caused by the lack of knowledge of the project manager about different solutions to different situations with different people.
- **Appointed Team:** The assumption that any group of people can immediately become an effective team is false and is rooted in ignorance.
- **Glass Case Plan:** The false assumption that the project plan guarantees project success is rooted in ignorance.
- **Inflexible Plan:** Change is a component present in all aspects of life and software projects are not excluded from the rule; thus, change must be accounted for in the project plan and failure in recognizing this often comes from ignorance.
- **Irrational Management:** A project manager without management experience or training is likely to lead to irrational management decisions, indecisiveness, among other negative management habits.
- **Leader Not Manager:** A person can be a great leader, but that does not automatically guarantee that they will also be a great manager [Laplante et al., 2005]. The lack of knowledge in project management is the major obstacle in this case.
- **Micro-Management:** This anti-pattern is often caused by a lack of knowledge about people management by the project manager. The micro-management behavior becomes an attempt in mitigating the manager's weaknesses, and is characterized by

over management of certain project aspects, aspects that are under the responsibility of their subordinates.

- **Mushroom Management:** The main, flawed assumption in this anti-pattern is that requirements are stable and well understood by both end users and the software team at project inception; developers are thus kept isolated from end users by management. We here assume that people generally make mistakes with good intentions in mind, and thus, we believe the major root cause for this anti-pattern is ignorance on the part of the project manager.
- **Project Mismanagement:** A project manager without management experience or training is likely to result in a software project that lacks proper monitoring and controlling.
- **Proletariat Hero:** A mistaken assumption by management that coercion is an efficient way to increase productivity and rooted in ignorance on the part of the project manager.
- **Rising Upstart:** This anti-pattern is caused by a project manager who does not know how to deal with a *Rising Upstart* – a "superstar" who cannot wait their time.
- **Size Isn't Everything:** The mistaken assumption that developers are interchangeable and that the number of people working on a project is inversely proportional to development time is deeply rooted in ignorance.
- **The Brawl:** This anti-pattern is characterized by a project manager with no leadership or management experience and knowledge; also caused by ignorance.
- **The Domino Effect:** "Treating resources for different projects in a collective manner, blurring project boundaries" [Brown et al., 2000] is a clear sign of lack of understanding by the project manager of the consequences of such actions.

It means that an impressive 64% of the anti-patterns in the consolidated list are due to ignorance on the part of the project manager, making it by far the most common root cause in our analysis.

5.3.3.4 Pride

There are three anti-patterns rooted in pride:

- **Micro-Management:** One of the root causes for a manager to become a micro-manager is pride in controlling all aspects of a project, including aspects that are under the responsibility of their subordinates.
- **Myopic Delivery:** In this anti-pattern, management's myopic view of the schedule is rooted in pride, leading to the refusal to accept changes to the original schedule, even when reducing staff or funding.
- **Ultimate Weapon:** Ultimate Weapons are extremely talented individuals in a software team who "can do great things, but often they know how great they are and their arrogance and self-righteousness can be just as great" [Laplante et al., 2005]. The difficulty in managing such "ultimate weapons" is in the pride of involved parties.

It means that approximately 14% of the anti-patterns in the consolidated list are due to pride on the part of the project manager, making it one of the three least common root causes in our analysis.

5.3.3.5 Sloth

There are six anti-patterns rooted in sloth:

- **Absentee Manager:** The absenteeism of the project manager is due to a lack of involvement and sense of responsibility on their part, which ultimately can be traced to sloth.
- **Inflexible Plan:** Change is a component present in all aspects of life and software projects are not excluded from the rule; thus, change must be accounted for in the project plan. Sloth in doing a carefully thought out plan leads to the generation of a plan that is not flexible to changes.
- **Process Disintegration:** "Despite its name this anti-pattern is primarily a people-caused anti-pattern that has process consequences" [Brown et al., 2000]. When processes begin to fail, the problem is often not with the process itself, but with the attitude of the people performing them; the root cause being sloth.
- **Project Mismanagement:** Sloth on the part of the project manager can lead to a software project that lacks proper monitoring and controlling.
- **Road to Nowhere:** There is hardly an explanation for carrying out a software project nowadays without planning, except when an acute case of sloth is present.
- **Ultimate Weapon:** Ultimate Weapons are extremely talented individuals in a software team and relying heavily on them can be a sign of sloth by the project manager, who is resorting to a simple and easy solution.

It means that 27% of the anti-patterns in the consolidated list are due to sloth on the part of the project manager.

5.3.3.6 Summary

A summary of these numbers is shown in Table 10; the most common root cause being *Ignorance* and the least common root cause being *Haste*. Note that given how some anti-patterns have more than one root cause, the sum of the percentages will not be 100%.

A close look at Table 9 shown earlier, reveals an interesting (and embarrassing!) phenomenon: 18 out of the 22 anti-patterns in our consolidated list have their roots in either *Ignorance* or *Sloth* on the part of the project manager.

This is confirmed by the results in Table 11, where the root causes *Ignorance* and *Sloth* are dissected further. In other words, *Ignorance* and *Sloth* alone account for approximately 82% of software project management anti-patterns.

Root Cause	**Number of Anti-Patterns**	**Percentage (out of 22 Anti-Patterns)**
Ignorance	14	64%
Sloth	6	27%
Pride	3	14%
Avarice	2	9%
Haste	1	5%

Table 10: Most common root causes of software project management anti-patterns.

Root Cause	**Number of Anti-Patterns**	**Percentage (out of 22 Anti-Patterns)**
Ignorance -only	12	55%
Sloth -only	4	18%
Both Ignorance *and* Sloth	2	9%
Total for Ignorance *and/or* Sloth	18	82%

Table 11: Analysis of *ignorance* and *sloth* root causes.

5.3.4 Categorization per Solution Type

Some of the anti-patterns in the consolidated list, in particular those that came from Brown, 1998, already had their solution types identified by the author. However, we replicated this analysis looking for the single major solution type among the four types described in section 5.2.4. For those anti-patterns that did not have this information, we identified their solution types based on the anti-pattern solution description provided by the author.

The final categorization is shown in Table 12. Note that while we have strived to find a single solution type, such as *Road to Nowhere* which requires a *Process* solution type, there are cases in which anti-patterns may require more than one solution type, such as *Inflexible Plan* which requires both *Process* and *Training*.

Also note that, as in previous sections, the table has been sorted by the criteria, in this case the solution type, not the anti-pattern name.

Anti-Pattern Name	**Solution Type**
Detailitis Plan	Process
Dry Waterhole	Process
Fire Drill	Process
Glass Case Plan	Process
Myopic Delivery	Process
Road to Nowhere	Process
Inflexible Plan	Process, Training
Absentee Manager	Role
Ultimate Weapon	Role
Micro-Management	Role, Training
Process Disintegration	Technology
All You Have Is a Hammer	Training
Appointed Team	Training
Irrational Management	Training
Leader Not Manager	Training
Mushroom Management	Training
Project Mismanagement	Training
Proletariat Hero	Training
Rising Upstart	Training
Size Isn't Everything	Training
The Brawl	Training
The Domino Effect	Training

Table 12: Software project management anti-patterns categorized by solution type.

5.3.4.1 Process

There are seven anti-patterns solved through a process type of solution:

- **Detailitis Plan:** At the core of the solution process, “the project plan should show primarily deliverables and should be supplemented with validation milestones.

Tracking is done on the estimated level of completeness, where completeness should be gross measurements rather than fine measurements" [Brown et al., 1998].

- **Dry Waterhole:** In order to avoid being excessively stringent about the job requirements or the candidate profile, the organization can improve its staff selection process by considering long-terms goals, aside from the usual short-term need to fill a given position [c2.com, 2014].
- **Fire Drill:** A Fire Drill scenario starts when management decides that development must progress immediately, often making unrealistic demands for software delivery. An effective solution that project management can implement is called *sheltering*. "In the sheltering solution, management creates and maintains two alternative project environments: internal and external; the majority of the software development staff operate in the internal environment, focused on making continual progress toward software delivery" [Brown et al., 1998].
- **Glass Case Plan:** The false assumption that the project plan guarantees project success is most effectively corrected through the creation of a project plan that "shows primarily deliverables and is supplemented with validation milestones. The deliverable plans should be updated weekly to ensure appropriate planning and controls that reduce project risks" [Brown et al., 1998].
- **Inflexible Plan:** Process is one of the two required solution types to inflexible plans. Change is a normal component in any software project and thus, it must be accounted for in the project plan, either through preemptive planning or reactive planning [Brown et al., 2000].
- **Myopic Delivery:** Management's refusal to accept changes to the original schedule, even when reducing staff or funding can be dealt with by following existing processes and practices. In other words, "stick to fundamental project management and system engineering principles" [Brown et al., 2000].
- **Road to Nowhere:** "As a manager and leader, you have to have a plan; without plan, work is undirected and without rationale. While there are a number of useful tools that can assist with project planning, they must not, however, be considered a substitute for a comprehensive plan or the planning activity itself" [Laplante et al., 2005].

It means that 32% of the anti-patterns in the consolidated list are solved through the pursuit of a process.

5.3.4.2 Role

There are three anti-patterns solved through a role type of solution:

- **Absentee Manager:** It is important to not make decisions for Absent Managers, which lets off the hook, and to attempt to create situations that force the Absent Managers to be present or that highlight their absence, such as in-person meetings [Laplante et al., 2005].
- **Micro-Management:** Role is one of the two required solution types to micro-management, characterized by over management of certain project aspects, aspects that are under the responsibility of subordinates. This anti-pattern can be mitigated by

clearly defining management roles; the project manager should not try to be the sole keeper of the problems that arise [Brown et al., 2000].

- **Ultimate Weapon:** Ultimate Weapons are extremely talented individuals in a software team, who, according to Laplante et al. (2005), "can do great things, but often they know how great they are and their arrogance and self-righteousness can be just as great. If these individuals are not managed effectively, the harm they cause can overshadow their achievements. The key is to diversify. This makes for a careful balancing act; you do not want to risk losing the good ones because you did not challenge them enough, but you must allow other team members to contribute".

It means that 14% of the anti-patterns in the consolidated list are solved by assigning responsibility to an individual or group.

5.3.4.3 Technology

There is one anti-pattern through a technology type of solution:

- **Process Disintegration:** "Despite its name this anti-pattern is primarily a people-caused anti-pattern that has process consequences" [Brown et al., 2000]. When processes begin to fail, the problem is often not with the process itself, but with the attitude of the people performing them. It is thus necessary to improve the morale of the development team; one way to achieve this is through an internal, collective effort, such as a web collaboration initiative, which can "reward individual creativity and imagination while also increasing people's inter-human connectivity" [Brown et al., 2000].

It means that 5% of the anti-patterns in the consolidated list are solved by the acquisition of a technology or product, making it the least common solution type in our analysis.

5.3.4.4 Training

There are thirteen anti-patterns through a training type of solution:

- **All You Have Is a Hammer:** "No methodology, process, model, or technique is appropriate in all situations. Rather, a sophisticated menu of alternatives is needed for each situation, application domain, and environment" [Laplante et al., 2005]. It is through training that one is able to gather this sophisticated menu of alternatives.
- **Appointed Team:** The most effective way to correct the mistaken assumption that any group of people can immediately become an effective team is through training.
- **Inflexible Plan:** Training is one of the two required solution types to inflexible plans. Change is a normal component in any software project and thus, "it must be recognized and accepted by the project manager, through a conceptual understanding of change, and a state of mind that enables flexibility and adaptability" [Brown et al., 2000].
- **Irrational Management:** A manager's inability to direct the development staff, characterized by irrational management decisions and indecisiveness can be mitigated through professional development and training.

- **Leader Not Manager:** "Inspiring leadership is very important, but so is effective management. The solution can be as simple as encouraging professional development in the areas of budgeting, organizational management, and planning" [Laplante et al., 2005].
- **Micro-Management:** Training is one of the two required solution types to micro-management. This anti-pattern often begins with a lack of knowledge about people management by the project manager; the micro-management behavior then becomes an attempt in mitigating the manager's weaknesses, and is characterized by over management of certain project aspects, aspects that are under the responsibility of their subordinates [Brown et al., 2000].
- **Mushroom Management:** There is a flawed assumption that requirements are stable and well understood by both end users and the software team at project inception, ultimately leading to poor design decisions. The most effective method of eliminating this mistaken assumption is to better prepare project managers for the position.
- **Project Mismanagement:** This anti-pattern is characterized by a software project that lacks proper monitoring and controlling. "Proper risk management is an effective means of resolving predictable symptoms and consequences of the Project Mismanagement anti-pattern. Risks are categorized in several useful ways: managerial, common project failure points, and quality. It's necessary to understand these categories in order to better qualify the risks" [Brown et al., 1998]. It is through training that one is able to obtain a solid understanding about risk management.
- **Proletariat Hero:** There is a mistaken assumption by management that coercion is an efficient way to increase productivity, and that people should be treated like machines. Training provides awareness that "harnessing and encouraging the creativity, innovation, and individuality within the workforce results in people taking pride in their work and being motivated to succeed both personally and collectively" [Laplante et al., 2005].
- **Rising Upstart:** "Rising Upstarts are superstars that cannot wait their time and want to forego the requisite time to learn, mature, and find their place. The best strategy is effective mentoring" [Laplante et al., 2005]. Managers need to be well-trained in order to be able to identify these Upstarts, recognize their potential, and work with them on a plan to develop their careers carefully.
- **Size Isn't Everything:** The mistaken assumption that developers are interchangeable and that the number of people working on a project is inversely proportional to development time can be most effectively resolved through training, leveraging off the works of others who have been in similar scenarios before.
- **The Brawl:** A project manager with no leadership or management experience and knowledge can be mitigated through professional development and training.
- **The Domino Effect:** "Treating resources for different projects in a collective manner, blurring project boundaries" [Brown et al., 2000] is a clear sign of lack of understanding by the project manager of the consequences of such actions, which can be remedied through training.

It means that an impressive 59% of the anti-patterns in the consolidated list are solved staff training and professional development, making it by far the most common solution type in our analysis.

5.3.4.5 Summary

A summary of these numbers is shown in Table 13; the common solution type being *Training* and the least common solution type being *Technology*. Note that given how some anti-patterns require more than one solution type, the sum of the percentages will not be 100%.

Table 14 further dissects the solution types *Training* and *Process*, showing that a total of 19 out of 22 anti-patterns, or 86%, can be addressed with at least one of these two solutions types.

Solution Type	**Number of Anti-Patterns**	**Percentage (out of 22 Anti-Patterns)**
Training	13	59%
Process	7	32%
Role	3	14%
Technology	1	5%

Table 13: Most common solution types for software project management anti-patterns.

Solution Type	**Number of Anti-Patterns**	**Percentage (out of 22 Anti-Patterns)**
Training -only	12	55%
Process -only	6	27%
Both Training *and* Process	1	5%
Total for Training *and/or* Process	19	86%

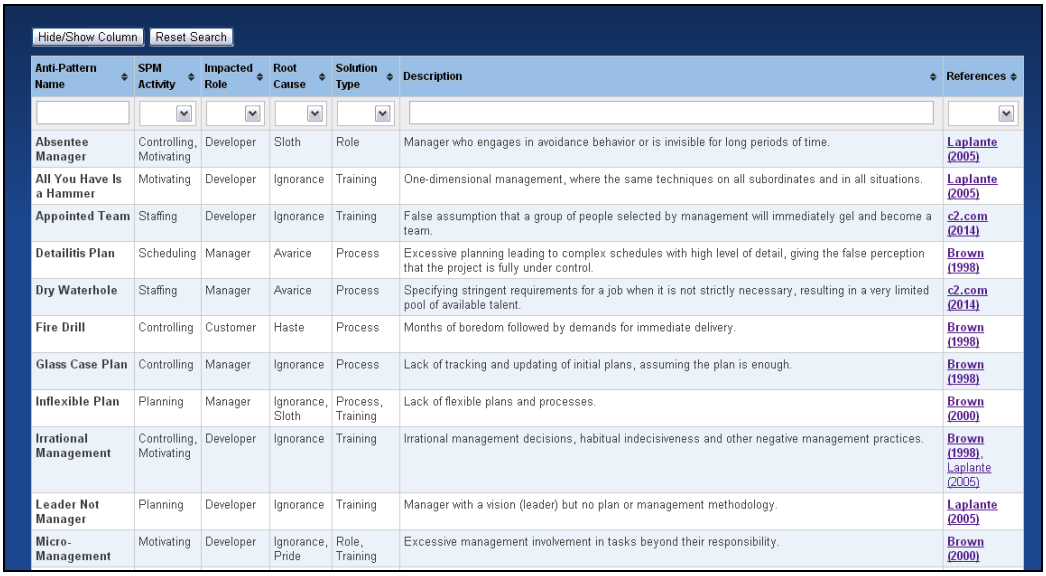
Table 14: Analysis of *training* and *process* solution types.

5.4 *Categorization Tool*

We have created a categorization tool in this thesis in order to make the results of this work more readily available and hopefully to have a more practical impact on the software development community. The tool can be found in the following address:

http://is.ls.fi.upm.es/research/spmantipatterns/home.html

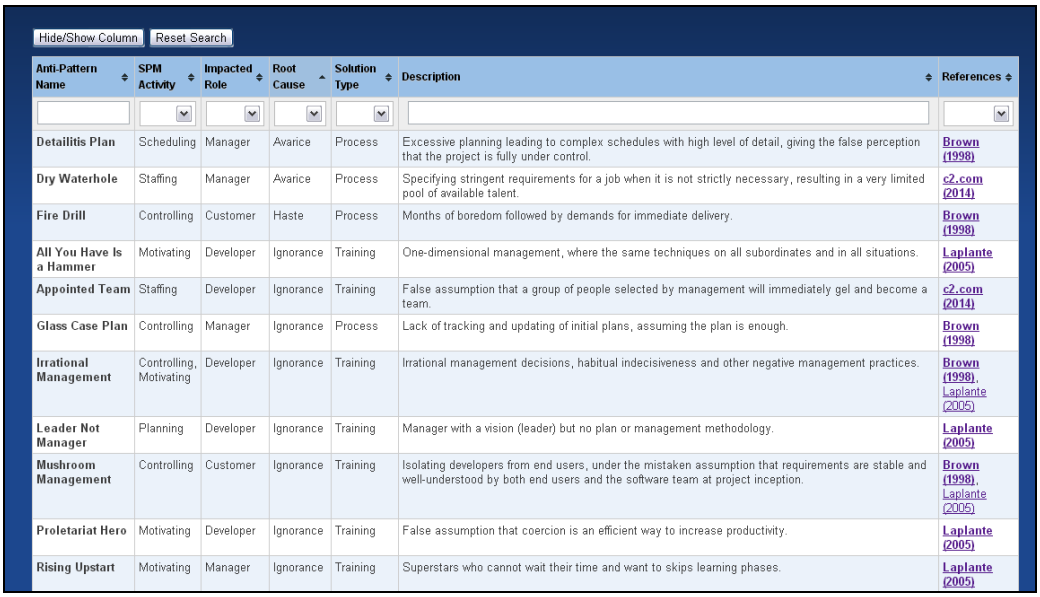
In essence, the tool is a website about anti-patterns in software project management that contains a comprehensive table of the results obtained in our anti-pattern categorization. The website was developed using a jQuery plug-in named *tablesorter*, and a sample template found on the Internet. The most relevant aspect of the table is the fact that it allows us to visualize different information about each anti-pattern simultaneously, as shown in Figure 1. In other words, for each anti-pattern, the table simultaneously shows seven variables that would otherwise be difficult to grasp: the anti-pattern name, the four categorization criteria (most impacted software project management activity, most impacted role, root cause, and solution type), the anti-pattern description, and the corresponding references.


Hide/Show Column | Reset Search

Anti-Pattern Name	SPM Activity	Impacted Role	Root Cause	Solution Type	Description	References
Absentee Manager	Controlling, Motivating	Developer	Sloth	Role	Manager who engages in avoidance behavior or is invisible for long periods of time.	Laplante (2005)
All You Have Is a Hammer	Motivating	Developer	Ignorance	Training	One-dimensional management, where the same techniques on all subordinates and in all situations.	Laplante (2005)
Appointed Team	Staffing	Developer	Ignorance	Training	False assumption that a group of people selected by management will immediately gel and become a team.	c2.com (2014)
Detailitis Plan	Scheduling	Manager	Avarice	Process	Excessive planning leading to complex schedules with high level of detail, giving the false perception that the project is fully under control.	Brown (1998)
Dry Waterhole	Staffing	Manager	Avarice	Process	Specifying stringent requirements for a job when it is not strictly necessary, resulting in a very limited pool of available talent.	c2.com (2014)
Fire Drill	Controlling	Customer	Haste	Process	Months of boredom followed by demands for immediate delivery.	Brown (1998)
Glass Case Plan	Controlling	Manager	Ignorance	Process	Lack of tracking and updating of initial plans, assuming the plan is enough.	Brown (1998)
Inflexible Plan	Planning	Manager	Ignorance, Sloth	Process, Training	Lack of flexible plans and processes.	Brown (2000)
Irrational Management	Controlling, Motivating	Developer	Ignorance	Training	Irrational management decisions, habitual indecisiveness and other negative management practices.	Brown (1998), Laplante (2005)
Leader Not Manager	Planning	Developer	Ignorance	Training	Manager with a vision (leader) but no plan or management methodology.	Laplante (2005)
Micro-Management	Motivating	Developer	Ignorance, Pride	Role, Training	Excessive management involvement in tasks beyond their responsibility.	Brown (2000)

Figure 1: Categorization tool – Initial table.

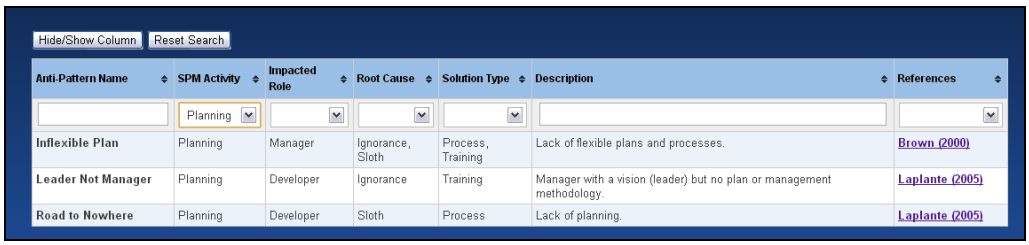
The table has three interesting features that might aggregate significant value to the user. First, the user is able to sort the list by any of the variables by simply clicking on the desired column header. Figure 2 shows the table sorted by *root cause*.



Anti-Pattern Name	SPM Activity	Impacted Role	Root Cause	Solution Type	Description	References
Detailitis Plan	Scheduling	Manager	Avarice	Process	Excessive planning leading to complex schedules with high level of detail, giving the false perception that the project is fully under control.	Brown (1998)
Dry Waterhole	Staffing	Manager	Avarice	Process	Specifying stringent requirements for a job when it is not strictly necessary, resulting in a very limited pool of available talent.	c2.com (2014)
Fire Drill	Controlling	Customer	Haste	Process	Months of boredom followed by demands for immediate delivery.	Brown (1998)
All You Have Is a Hammer	Motivating	Developer	Ignorance	Training	One-dimensional management, where the same techniques on all subordinates and in all situations.	Laplante (2005)
Appointed Team	Staffing	Developer	Ignorance	Training	False assumption that a group of people selected by management will immediately gel and become a team.	c2.com (2014)
Glass Case Plan	Controlling	Manager	Ignorance	Process	Lack of tracking and updating of initial plans, assuming the plan is enough.	Brown (1998)
Irrational Management	Controlling, Motivating	Developer	Ignorance	Training	Irrational management decisions, habitual indecisiveness and other negative management practices.	Brown (1998), Laplante (2005)
Leader Not Manager	Planning	Developer	Ignorance	Training	Manager with a vision (leader) but no plan or management methodology.	Laplante (2005)
Mushroom Management	Controlling	Customer	Ignorance	Training	Isolating developers from end users, under the mistaken assumption that requirements are stable and well-understood by both end users and the software team at project inception.	Brown (1998), Laplante (2005)
Proletariat Hero	Motivating	Developer	Ignorance	Training	False assumption that coercion is an efficient way to increase productivity.	Laplante (2005)
Rising Upstart	Motivating	Manager	Ignorance	Training	Superstars who cannot wait their time and want to skips learning phases.	Laplante (2005)

Figure 2: Categorization tool – Table sorted by root cause.

Second, the tool allows the user to filter columns by selecting a value from the drop-down list in each column header. For example, to view all anti-patterns related to the software project management activity of *Planning*, the user just needs to select "Planning" from the drop-down list in that column header and the table will filter out any anti-pattern that does not match it, as shown in Figure 3. To clear the filter, the user can click on the "Reset Search" button.

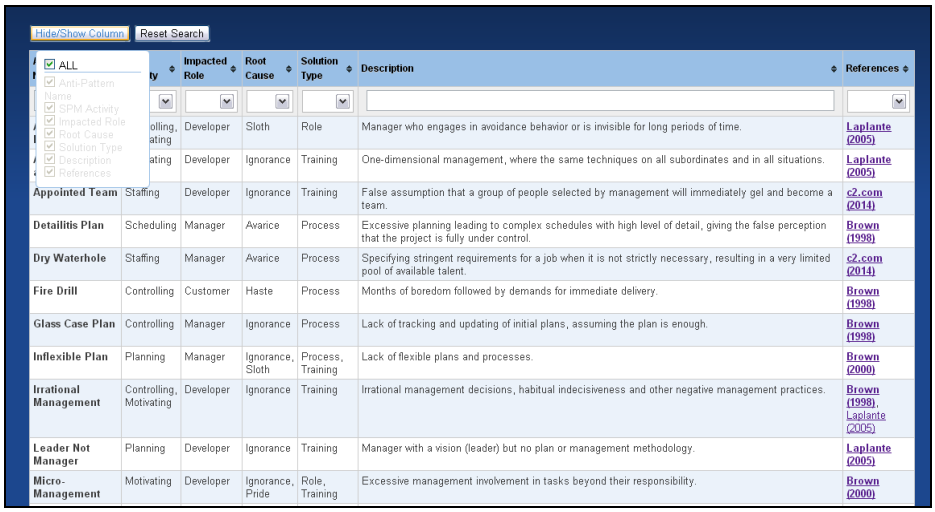


Anti-Pattern Name	SPM Activity	Impacted Role	Root Cause	Solution Type	Description	References
	Planning					
Inflexible Plan	Planning	Manager	Ignorance, Sloth	Process, Training	Lack of flexible plans and processes.	Brown (2000)
Leader Not Manager	Planning	Developer	Ignorance	Training	Manager with a vision (leader) but no plan or management methodology.	Laplante (2005)
Road to Nowhere	Planning	Developer	Sloth	Process	Lack of planning.	Laplante (2005)

Figure 3: Categorization tool – Table filtered by the SPM activity *Planning*.

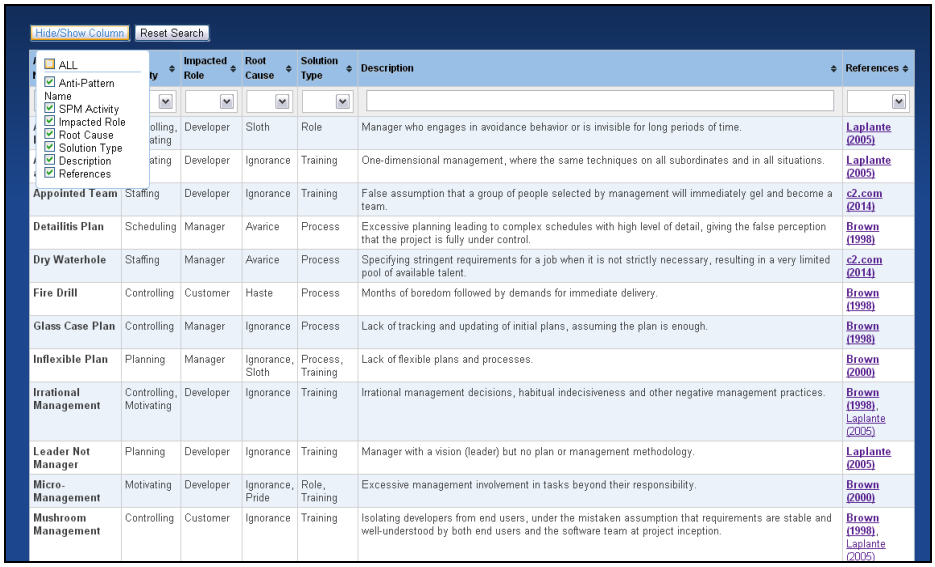
And finally, the user has the ability of hiding columns. This feature is especially useful if the user is not interested in all of the variables, allowing thus the user to focus on the important few.

Hiding columns requires three simple steps. Figure 4 shows the first one: the user must click on the "Show/Hide Column" button on the top left-hand-side corner of the screen. Figure 5 shows the second step: de-selecting the "ALL" checkbox. Figure 6 shows the last step: by de-selecting the checkbox corresponding to a given column, the user is able to hide that column; the inverse is also true, re-selecting the corresponding checkbox un-hides the column. In the example show here, we have hidden the *description* column.



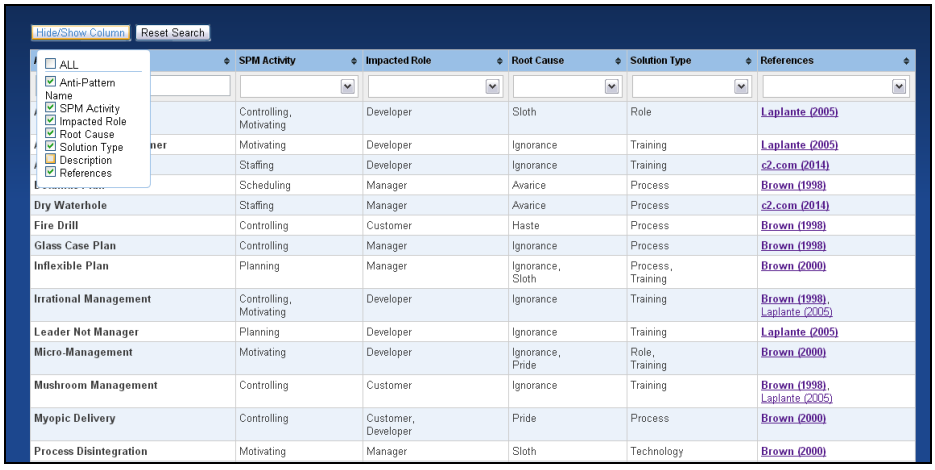
Anti-Pattern Name	SPM Activity	Impacted Role	Root Cause	Solution Type	Description	References
	olling, ating	Developer	Sloth	Role	Manager who engages in avoidance behavior or is invisible for long periods of time.	Laplante (2005)
	ating	Developer	Ignorance	Training	One-dimensional management, where the same techniques on all subordinates and in all situations.	Laplante (2005)
Appointed Team	Staffing	Developer	Ignorance	Training	False assumption that a group of people selected by management will immediately gel and become a team.	c2.com (2014)
Detailitis Plan	Scheduling	Manager	Avarice	Process	Excessive planning leading to complex schedules with high level of detail, giving the false perception that the project is fully under control.	Brown (1998)
Dry Waterhole	Staffing	Manager	Avarice	Process	Specifying stringent requirements for a job when it is not strictly necessary, resulting in a very limited pool of available talent.	c2.com (2014)
Fire Drill	Controlling	Customer	Haste	Process	Months of boredom followed by demands for immediate delivery.	Brown (1998)
Glass Case Plan	Controlling	Manager	Ignorance	Process	Lack of tracking and updating of initial plans, assuming the plan is enough.	Brown (1998)
Inflexible Plan	Planning	Manager	Ignorance, Sloth	Process, Training	Lack of flexible plans and processes.	Brown (2000)
Irrational Management	Controlling, Motivating	Developer	Ignorance	Training	Irrational management decisions, habitual indecisiveness and other negative management practices.	Brown (1998), Laplante (2005)
Leader Not Manager	Planning	Developer	Ignorance	Training	Manager with a vision (leader) but no plan or management methodology.	Laplante (2005)
Micro-Management	Motivating	Developer	Ignorance, Pride	Role, Training	Excessive management involvement in tasks beyond their responsibility.	Brown (2000)

Figure 4: Categorization tool – Hiding columns (Part 1 of 3).



Anti-Pattern Name	SPM Activity	Impacted Role	Root Cause	Solution Type	Description	References
	olling, ating	Developer	Sloth	Role	Manager who engages in avoidance behavior or is invisible for long periods of time.	Laplante (2005)
	ating	Developer	Ignorance	Training	One-dimensional management, where the same techniques on all subordinates and in all situations.	Laplante (2005)
Appointed Team	Staffing	Developer	Ignorance	Training	False assumption that a group of people selected by management will immediately gel and become a team.	c2.com (2014)
Detailitis Plan	Scheduling	Manager	Avarice	Process	Excessive planning leading to complex schedules with high level of detail, giving the false perception that the project is fully under control.	Brown (1998)
Dry Waterhole	Staffing	Manager	Avarice	Process	Specifying stringent requirements for a job when it is not strictly necessary, resulting in a very limited pool of available talent.	c2.com (2014)
Fire Drill	Controlling	Customer	Haste	Process	Months of boredom followed by demands for immediate delivery.	Brown (1998)
Glass Case Plan	Controlling	Manager	Ignorance	Process	Lack of tracking and updating of initial plans, assuming the plan is enough.	Brown (1998)
Inflexible Plan	Planning	Manager	Ignorance, Sloth	Process, Training	Lack of flexible plans and processes.	Brown (2000)
Irrational Management	Controlling, Motivating	Developer	Ignorance	Training	Irrational management decisions, habitual indecisiveness and other negative management practices.	Brown (1998), Laplante (2005)
Leader Not Manager	Planning	Developer	Ignorance	Training	Manager with a vision (leader) but no plan or management methodology.	Laplante (2005)
Micro-Management	Motivating	Developer	Ignorance, Pride	Role, Training	Excessive management involvement in tasks beyond their responsibility.	Brown (2000)
Mushroom Management	Controlling	Customer	Ignorance	Training	Isolating developers from end users, under the mistaken assumption that requirements are stable and well-understood by both end users and the software team at project inception.	Brown (1998), Laplante (2005)

Figure 5: Categorization tool – Hiding columns (Part 2 of 3).



Anti-Pattern Name	SPM Activity	Impacted Role	Root Cause	Solution Type	References
	Controlling, Motivating	Developer	Sloth	Role	Laplante (2005)
ner	Motivating	Developer	Ignorance	Training	Laplante (2005)
	Staffing	Developer	Ignorance	Training	c2.com (2014)
	Scheduling	Manager	Avarice	Process	Brown (1998)
Dry Waterhole	Staffing	Manager	Avarice	Process	c2.com (2014)
Fire Drill	Controlling	Customer	Haste	Process	Brown (1998)
Glass Case Plan	Controlling	Manager	Ignorance	Process	Brown (1998)
Inflexible Plan	Planning	Manager	Ignorance, Sloth	Process, Training	Brown (2000)
Irrational Management	Controlling, Motivating	Developer	Ignorance	Training	Brown (1998), Laplante (2005)
Leader Not Manager	Planning	Developer	Ignorance	Training	Laplante (2005)
Micro-Management	Motivating	Developer	Ignorance, Pride	Role, Training	Brown (2000)
Mushroom Management	Controlling	Customer	Ignorance	Training	Brown (1998), Laplante (2005)
Myopic Delivery	Controlling	Customer, Developer	Pride	Process	Brown (2000)
Process Disintegration	Motivating	Manager	Sloth	Technology	Brown (2000)

Figure 6: Categorization tool – Hiding columns (Part 3 of 3).

6 Conclusions

In this thesis, we have attempted to answer our two research questions:

- What are the different anti-patterns in software project management?
- How can these anti-patterns be categorized?

We have reviewed the available literature in search of project management anti-patterns that were relevant to the five software project management activities identified by Kerzner (2013) and Peters (2008). From our research, we were able to generate a consolidated list of twenty-two software project management anti-patterns, shown earlier in Chapter 4. We then categorized these anti-patterns using the four selected criteria discussed in Chapter 5. In addition, we have developed a website, also presented in Chapter 5, in order to display the results of our analysis in a way that is easier to understand, as well as to make the results more readily accessible.

The categorization of software project management anti-patterns carried out in this work has led to a number of interesting findings, which we will discuss next in further detail.

The analysis of the software project management activities most impacted by the anti-patterns in our consolidated list has generated what we believe to be expected results. The activities most impacted ended up being those that last the longest throughout a software development project: controlling and motivating. Both activities are, in fact, active throughout the entire software project [Peters, 2008] so these results should come as no shocking revelation, given the higher rate of exposure of the two activities to potential anti-patterns.

Investigating the roles most impacted by the anti-patterns in our consolidated list has led to a paradoxical discovery. In nearly half of the anti-patterns we have analyzed in this work, it was the role of the manager that suffered the greatest impact; managers who in many cases, and for a variety of reasons ranging from lack of experience to ambition and greed, had provoked themselves the situation they were suffering from. It is an ironic revelation that should serve as motivation for project managers to avoid software project management malpractices and to engage in continuous improvement and professional development.

Another noteworthy finding is that four anti-patterns significantly impact the role of the customer: *Fire Drill*, *Mushroom Management*, *Myopic Delivery*, and *Project Mismanagement*. While all software project management anti-patterns represent undesirable scenarios, anti-patterns that have a significant impact on the customer should not be acceptable.

Schedule pressure, a manifestation of haste, is a common feature of software projects, occurring in about 75 percent of all medium-sized projects and in 90 percent or more of all large projects [McConnell, 2004]. Haste leads to unrealistic schedules and expectations from management, and while it certainly plays an important role in software project failure, it is interesting to note that haste is the least common root cause of the software project management anti-patterns in our consolidated list. In fact, the most common root cause is ignorance, materialized by the project manager's lack of experience or training. It might just be, after all, that the explanation for a project manager's hasty decisions or actions is simply a lack of preparation to successfully fulfill the position. This should be interpreted in a constructive manner, as it is something that can be remedied, and the solution should not be to terminate underperforming project managers and to hire new ones with potentially similar

limitations. According to Laplante et al. (2005), "Effective leaders are in short supply so discarding them is usually not an option." We believe a much more powerful and long-lasting solution is to invest on these managers, training current managers to enhance their skills and preparing aspiring managers for the future.

Another interesting, yet alarming discovery is that the majority of software project management anti-patterns in our consolidated list can be solved by staff training, and in particular, project management training. In other words, staff is being assigned to project management without being fully qualified or trained for the position [McConnell, 1998]. This scenario typically occurs when a staff member, such as an engineer or programmer, is promoted into a leadership or project management position, often without proper training and preparation. Given that the software project manager has a greater impact on project success than all other factors combined [Weinberg, 1994], effective leadership and management are essential in any software project. We believe this to be one of the most critical issues in the software industry today.

It is important to keep in mind, while reading these findings, some of the limitations of this work, including the very limited number of sources in the topic of anti-patterns. As discussed earlier in Chapter 3, we were not able to find academic papers with in-depth analysis of anti-patterns, and only three sources exist in literature about this topic plus one online source. This has certainly been the most challenging aspect of conducting a thesis on this topic. Another important limitation is that, even though the anti-pattern categorization was done based on the characteristics of each anti-pattern and with an honest intent to categorize them to the best of our abilities, experience, and knowledge, there is still a human factor at play in our judgment that can introduce a significant margin of error in the analysis.

Perhaps an extension to this work could be to conduct an anti-pattern experiment to validate, or correct, the results presented here. The advantage of investigating software project management anti-patterns is that they do not manifest themselves in corporations only, which makes them easier to be observed, and thus studied, and hopefully mitigated. Managing projects and developers in a field as young and as complex as that of software engineering is no easy feat and has been compared to "herding cats" [Peters, 2008]. Any organization or development group, of any size, can suffer from such anti-patterns.

A second potential extension to this work could be an analysis of how to solve or to avoid software project management anti-patterns. The anti-pattern sources used in this work would be a recommended starting point; the authors offer at least one re-factored solution per anti-pattern. Additional, effective techniques are likely to exist, and thus, further investigation would be needed.

A final potential way in which this work could be extended is through an investigation of whether all software project management anti-patterns have a re-factored solution or not. As far as the anti-pattern sources consulted for this work go, all of the anti-patterns examined have been accompanied by a corresponding re-factored solution. However, even though the works of these authors are extensive, we do not believe all existing software project management anti-patterns have been identified yet. Consequently, it opens a door to the possibility of anti-patterns that simply do not have a re-factored solution.

A few decades ago, there was an assumption that we had effective software project managers [Boehm, 1981]. This assumption has not been confirmed over the years. As pointed out by Brown et al. (1998), Laplante et al. (2005), McConnell (1998), and Peters (2008), effective software project managers are a scarce commodity in the software industry. There are

several software project management anti-patterns that can obstruct the path of someone becoming an effective project manager. However, while not all anti-patterns can be avoided as some are outside the control of the project manager, those anti-patterns that *are* in control should and can be avoided.

Although solving or avoiding software project management anti-patterns is outside the scope of this thesis, we believe the results of the analysis carried out here to be relevant to both seasoned and inexperienced project managers. A project manager, for example, who is observing recurrent project schedule slips, can access the anti-pattern categorization website to look for anti-patterns related to *Scheduling*, and find references to re-factored solutions that can combat such negative scenarios. Inexperienced project managers, for example, who are struggling with managing their first software projects, can use the categorization website to look for anti-patterns related to *Ignorance*, and find references to re-factored solutions that can help them identify where they might be failing and also mitigate the consequences of the anti-pattern scenario.

It is the practical use of the results of this thesis what I have found to be most rewarding about this work. The topic of anti-patterns in software project management allowed me to study recurrent malpractices that are especially relevant in today's software-driven world. To think that others might hopefully benefit from this work, by using it to avoid common software project management malpractices, has served as a true motivation factor throughout this research.

This work is of particular interest to me as I have grown increasingly interested over the years in the management of people. Not the boss-subordinate management relationship that is often seen in the industry and in popular culture, but the type of management that is able to create the right environment and conditions for an individual and for a team to be able to unlock their full potential. Simply speaking, it means trading the satisfaction of reaching a solution or accomplishing a goal by that of helping others to do it instead. In the words credited to an ancient Chinese philosopher, "A leader is best when people barely know he exists, when his work is done, his aim fulfilled, and they will say: we did it ourselves."

7 References

[Akroyd, 1996] Akroyd, M. *AntiPatterns: Vaccinations against Object Misuse*. Object World West Conference, San Francisco, CA, 1996.

[Boehm, 1981] Boehm, B.W. *Software Engineering Economics*. Englewood Cliffs, NJ: Prentice-Hall, 1981.

[Brooks, 1979] Brooks, F. *The Mythical Man-Month: Essays on Software Engineering.* Reading, MA: Addison-Wesley, 1979.

[Brown et al., 1998] Brown, W.J., Malveau, R.C., McCormick, H.W., Mowbray, T.J. *AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis*. New York, NY: John Wiley & Sons, 1998.

[Brown et al., 2000] Brown, W.J., McCormick, H.W., Thomas, S.W. *AntiPatterns in Project Management*. New York, NY: John Wiley & Sons, 2000.

[c2.com, 2014] *C2.com*. Portland Pattern Repository's Wiki: Management Anti Pattern Road Map, 2014. Web. 23 May 2014.

[Coplien, 1995] Coplien, J. *A Development Process Generative Pattern Language*. PLoP, 1995.

[Gamma et al., 1994] Gamma, E., Helm, R., Johnson, R., Vlissides, J. *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading, MA: Addison-Wesley, 1994.

[Kerzner, 2013] Kerzner, H.R. *Project Management: A Systems Approach to Planning, Scheduling, and Controlling* (11th ed.). New York, NY: John Wiley & Sons, 2013.

[Koenig, 1995] Koenig, A. Patterns and Antipatterns. *Journal of Object-Oriented Programming*, 1995.

[Laplante et al., 2005] Laplante, P.A., Neill, C.J. *Antipatterns: Identification, Refactoring, and Management*. Boca Raton, FL: Taylor & Francis, 2005.

[Merriam-Webster, 2014] "training." *Merriam-Webster.com*. Merriam-Webster, 2014. Web. 23 May 2014.

[McConnell, 1998] McConnell, S. *Software Project Survival Guide: How to Be You're your First Important Project Isn't Your Last*. Redmond, WA: Microsoft Press, 1998.

[McConnell, 2004] McConnell, S. *Professional Software Development: Shorter Schedules, Higher Quality Products, More Successful Projects, Enhanced Careers*. Boston, MA: Addison-Wesley, 2004.

[McConnell, 2006] McConnell, S. *Software Estimation: Demystifying the Black Art*. Redmond, WA: Microsoft Press, 2006.

[Peters, 2008] Peters, L.J. *Getting Results from Software Development Teams*. Redmond, WA: Microsoft Press, 2008.

[Peters, 2012] Peters, L.J. *Managing Software Development Teams*. Auburn, WA: Software Consultants International Limited, 2012.

[Stamelos, 2009] Stamelos, I. Software Project Management Anti-patterns. *The Journal of Systems and Software*, 2009.

[Webster, 1995] Webster, B.F. *Pitfalls of Object-Oriented Development*. New York, NY: M & T Books, 1995.

[Weinberg, 1994] Weinberg, G.M. *Quality Software Management, Volume 3: Congruent Action*. New York, NY: Dorset House Publishing, 1994.